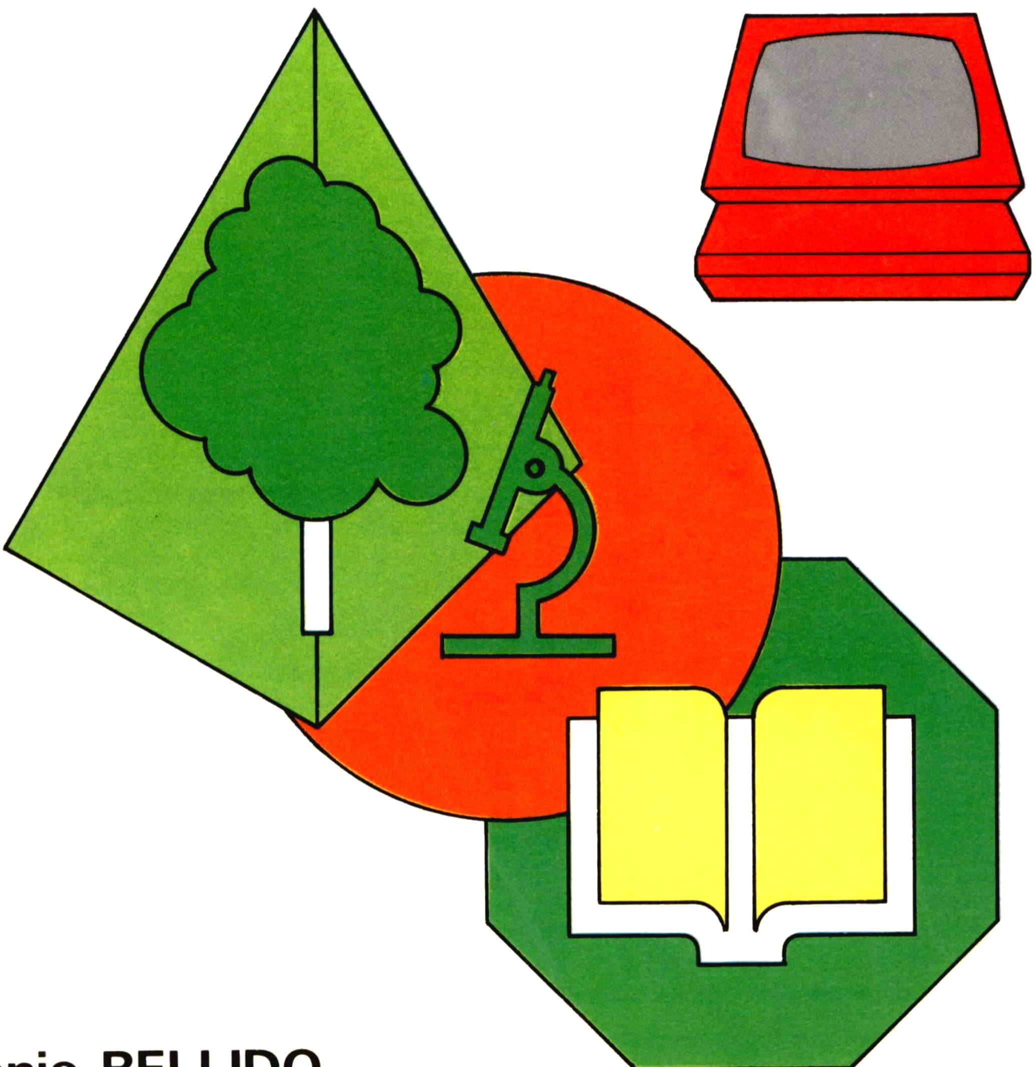
BASIC PARA MAESTROS

- Curso de programación basic
- Didáctica del basic
- Ejercicios y aplicaciones resueltos
- Introducción a la Informática



Antonio BELLIDO
Arsenio SANCHEZ

PARANINFO S.A.

BASIC para MAESTROS

- CURSO DE PROGRAMACION BASIC
- DIDACTICA DEL BASIC
- APLICACIONES Y EJERCICIOS RESUELTOS
- INTRODUCCION A LA INFORMATICA

Antonio Bellido y
Arsenio Sánchez

BASIC para MAESTROS

- CURSO DE PROGRAMACION BASIC
- DIDACTICA DEL BASIC
- APLICACIONES Y EJERCICIOS RESUELTOS
- INTRODUCCION A LA INFORMATICA

[illegible]
Sevilla, 19-11-85

1985

PARANINFO S.A.

MADRID

Este libro se complementa con la obra
BASIC PARA ESTUDIANTES
de los mismos autores.

IMPRESO EN ESPAÑA
PRINTED IN SPAIN

ISBN: 84-283-1375-X

Depósito Legal: M. 4.483-1985

PARANINFO S.A. Magallanes, 25 - 28015 MADRID (4-3463)

INDICE GENERAL

SENTENCIAS AUXILIARES:

APENDICES

SISTEMAS FISICOS

PROLOGO

Nadie puede negar el alto grado de desarrollo alcanzado por la Informática ni el importante papel que desempeña en la sociedad actual.

Sus imparables progresos y su enorme difusión nos llevan a reflexionar sobre sus posibilidades en el futuro inmediato y, nos agrade o no, llegamos a la conclusión de que, en pocos años, vamos a vivir en un mundo poderosamente informatizado en el que la sociedad va a ver absorbida, o quizás anulada, hasta la misma libertad personal de los individuos que permanezcan ajenos a ella.

Para evitarlo, es necesario esforzarse en conseguir que el mayor número posible de individuos acceda a los conocimientos informáticos, los asimile, los utilice y, en consecuencia, se beneficie de sus importantes avances tecnológicos.

De la misma forma que los niños se inician en el conocimiento de las Lenguas, las Matemáticas, la Expresión Plástica o la Expresión Corporal, y los perfeccionan a lo largo de su vida, es necesario que se inicien en la Informática, si no se quiere que gran parte de los futuros hombres queden marginados de ese amplio e importante campo del saber.

Hasta hace pocos años, la utilización de los ordenadores en el campo de la enseñanza estaba limitado, por sus elevados costes y mantenimiento, a grupos muy reducidos. En cambio, hoy, con la aparición de los microordenadores, su relativamente bajo coste en el mercado y, como consecuencia más inmediata, su difusión, resulta forzosa la incorporación de la Informática a los centros escolares con tres objetivos fundamentales:

1. Como poderoso auxiliar en la gestión del propio centro, para llevar la contabilidad, registrar las evaluaciones de los alumnos, organizar los ficheros, etc.
2. Como medio para guiar, apoyar y reforzar el aprendizaje de las diferentes materias del currículo escolar.
3. Como materia fundamental, por sí misma, de dicho currículo escolar.

Es en este último objetivo en el que, sobre todos los demás, conviene centrar los mayores esfuerzos para conseguir una cultura informática en su doble vertiente: la del ordenador y la de los lenguajes de programación.

Es necesario conocer el ordenador y saber manejarlo, ya que permite ser usado como un instrumento intelectual que, por tanto, sirve para desarrollar la lógica, la reflexión, el análisis, la síntesis...; en definitiva, las capacidades intelectuales, al tiempo que es un importante elemento inter y multidisciplinar.

Por otra parte, los lenguajes de programación –debido a los avances tecnológicos– han evolucionado tanto que cada vez están más cercanos al lenguaje natural. Y, así como el dominio de éste es un objetivo prioritario de todo proceso educativo, ha de serlo también el dominio de aquéllos para comunicarse en un futuro muy próximo.

Sin embargo, los lenguajes de programación presentan dificultades para los no iniciados, ya que, por su origen –casi siempre anglosajón–, por su forma de utilización y su función,

requiere un esfuerzo adicional para interpretar su vocabulario y comprender su sintaxis en quienes no lo conocen.

Esta publicación, dirigida a los profesores interesados en la Informática –que carecen de conocimientos de la misma o los poseen muy elementales–, pretende introducirlos en ella a través del lenguaje de programación BASIC, eliminando, dentro de sus limitaciones, los obstáculos citados y aportando sugerencias didácticas que posibiliten su aplicación en las aulas.

Para ello, la hemos estructurado en cuatro grandes apartados, de los que, a continuación ofrecemos una panorámica general.

En la INTRODUCCION explicamos algunos conceptos fundamentales en programación, tales como líneas de programa, algoritmo, constante, variable, etc.; cómo están configurados, en forma de fichas, los capítulos dedicados al lenguaje BASIC; una posible metodología para introducir las instrucciones o comandos, y cómo teclear un programa, además de sugerencias didácticas y ejercicios.

En el apartado de BASIC, exponemos el vocabulario de este lenguaje de programación, junto con su traducción española y su sintaxis; su interpretación y sus posibilidades de uso, así como sugerencias didácticas y ejercicios, con la solución de los más significativos.

En el apartado de APENDICES, figuran un esquema de los comandos BASIC, los diagramas de flujo, la organización de las pantallas, las funciones lógicas aplicadas, el troceado de cadenas, la tabla del código ASCII y los sistemas de numeración decimal, binario y hexadecimal utilizados en Informática.

Termina la publicación con el apartado de SISTEMAS FISICOS, en el que se da una visión somera del computador, las unidades de información, la memoria, los interfaces, los periféricos, los programas y el logical, para finalizar con unas reflexiones sobre el profesor, el alumno y el ordenador.

Recomendamos a los profesores leer previamente los epígrafes que se incluyen a partir de la página 173 y muy especialmente los referentes a "Los programas", "El lógical" y "Profesor, alumno y ordenador" ya que serán muy útiles para la comprensión de todo el libro y el desarrollo de la enseñanza.

Se cubre así un ciclo que permite al maestro no sólo conocer y dominar un lenguaje de programación, sino también explicarlo a sus alumnos.

Para aquellos profesores especialmente interesados en la enseñanza del BASIC a los más pequeños les recomendamos la lectura de las obras BASIC PARA NIÑOS y BASIC AVANZADO PARA NIÑOS, publicadas en esta misma editorial.

Para los mayores recomendamos el libro BASIC PARA ESTUDIANTES, editado también por PARANINFO, S.A.

Esperamos que la publicación cumpla nuestros deseos de utilidad y, desde luego, estamos abiertos para recibir y aceptar cuantas críticas y sugerencias tengan la amabilidad de hacernos los lectores.

LOS AUTORES

INTRODUCCION

Un **programa** es una secuencia de órdenes o instrucciones dadas al computador, que tiene por objetivo final la resolución de un problema.

Al escribir el programa, **el programador** establece esta secuencia numerando las líneas del programa para que el computador las almacene en su memoria en el orden establecido. Por ejemplo:

1Ø INPUT A$ (1)
2Ø PRINT A$
3Ø STOP

El computador almacenará en este orden las líneas 1Ø, 2Ø y 3Ø del programa, aunque se le hayan suministrado en otro, ya que los números de las líneas establecen la secuencia de éstas.

Para escribir el programa es necesario conocer **el lenguaje de programación** que se va a utilizar, y para introducirlo en la memoria del computador es necesario teclear sin errores cada una de sus líneas mediante el teclado del propio computador, por lo que es necesario conocer las normas que, al respecto, proporciona el MANUAL de cada ordenador o computador.

Cada *lenguaje de programación* está compuesto por un **vocabulario** y una **sintaxis** determinados.

El *vocabulario* es el conjunto de palabras que representan los diferentes **comandos** o **instrucciones** que acepta e interpreta el lenguaje de programación utilizado.

La *sintaxis* es el conjunto de normas que se deben seguir para escribir correctamente un programa.

De entre los muchos y variados tipos de lenguajes de programación que existen, hemos elegido el **BASIC** por tres razones:

1. Da prestaciones muy altas.
2. Es fácil de asimilar.
3. La mayoría de los microcomputadores lo llevan incorporado o, en todo caso, lo aceptan.

El **BASIC** tiene un *vocabulario* y una *sintaxis* que podemos considerar normalizados. No obstante, cada marca de ordenador suele imponer ciertas peculiaridades, que le son propias, al vocabulario y a la sintaxis del BASIC, que deben ser tenidos muy en cuenta por el usuario de ese ordenador. Por esta razón, el lector –en caso de tener su propia máquina– deberá comparar lo que aquí se escribe con los criterios que, sobre el mismo tema, mantenga el MANUAL de su ordenador.

---

(1) Los computadores utilizan Ø en vez de 0.

Para poder ceñirnos a un criterio, hemos utilizado una serie de instrucciones comunes a la mayoría de los ordenadores personales.

Con respecto a los modos de teclear un programa en un ordenador, diremos que existen dos:

A) Apretando todos y cada uno de los caracteres que conforman las líneas del programa. Tal es el caso en los ordenadores COMMODORE 64, ORIC, DRAGON, etc.

B) Apretando todos y cada uno de los caracteres que conforman las líneas del programa, excepto los comandos o instrucciones del BASIC, para los cuales basta con apretar la tecla donde figura la palabra BASIC correspondiente. El ejemplo típico lo tenemos en el SINCLAIR ZX-81 y SPECTRUM, ofreciendo una opción similar el MICRO-PROFFESSOR.

En todo lo que sigue nos referiremos exclusivamente a la programación en BASIC, sin tener en cuenta la forma en que se maneja el teclado y otros detalles que son específicos de cada computador, dejando al lector la tarea de aprender las especificaciones del microcomputador que vaya a utilizar en el MANUAL correspondiente.

Una vez escrito y tecleado el programa, habremos puesto una secuencia de órdenes en la memoria del ordenador pero, para hacer trabajar este programa, es decir, para hacer que el ordenador obedezca una tras otra las instrucciones de la secuencia, debemos **correrlo** o **ejecutarlo**. Para ello, el BASIC tiene el comando **RUN**, cuyo significado en español es, precisamente, *"correr"* o *"ejecutar"*.

Es muy posible que un programa escrito, tecleado y ejecutado por primera vez no funcione correctamente, ya que —casi con seguridad— habremos cometido algún tipo de error, que el propio computador se encargará de indicarnos.

La tarea de hacer funcionar correctamente un programa, con un razonable aprovechamiento de la memoria del computador, se llama **depurar el programa.** Este trabajo lo desarrollaremos en base a las facilidades de **edición** que nos ofrezca la máquina que estemos utilizando.

Cuando el programa funcione exactamente como su diseñador desea, éste debe documentarlo de tal forma que cualquier usuario del mismo pueda utilizarlo convenientemente.

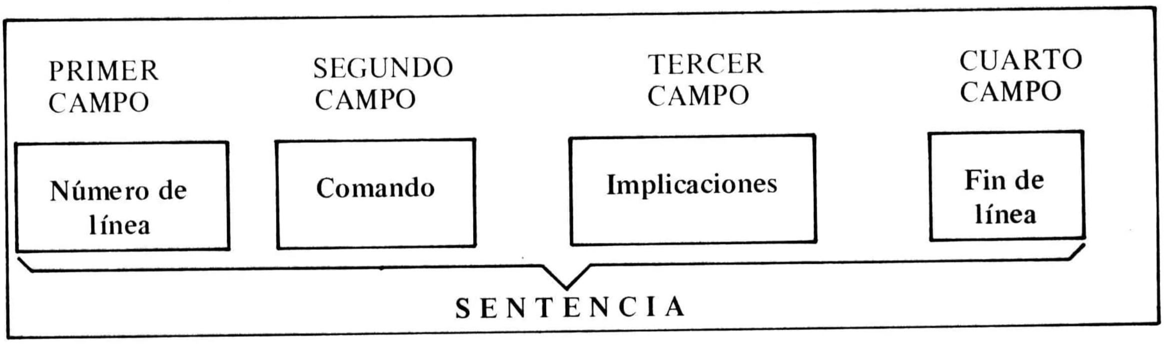
Para una mayor eficacia en el desarrollo de un programa, deben utilizarse con frecuencia **listados** del mismo, los cuales se obtendrán a través de los comandos oportunos, que harán un **volcado** del programa en una **impresora.** Gracias a esta máquina, obtendremos una copia sobre papel del programa que a la sazón esté en la memoria del computador.

Igualmente, es prudente **salvar** (guardar) el programa en cinta casete, microcinta o disco, para prevenir que, por cualquier razón (un fallo eléctrico, por ejemplo), perdamos el contenido de la memoria del computador. Si tal cosa ocurriese y el programa estuviese *salvado*, bastaría con proceder a **la carga** del programa desde el soporte de información externo (cinta, disco, etc.) a la memoria del ordenador.

Este conjunto de actividades —que aquí se han visto muy someramente— determinan la actividad de un programador. Poco a poco, el lector irá profundizando en ellas a medida que aumente su experiencia.

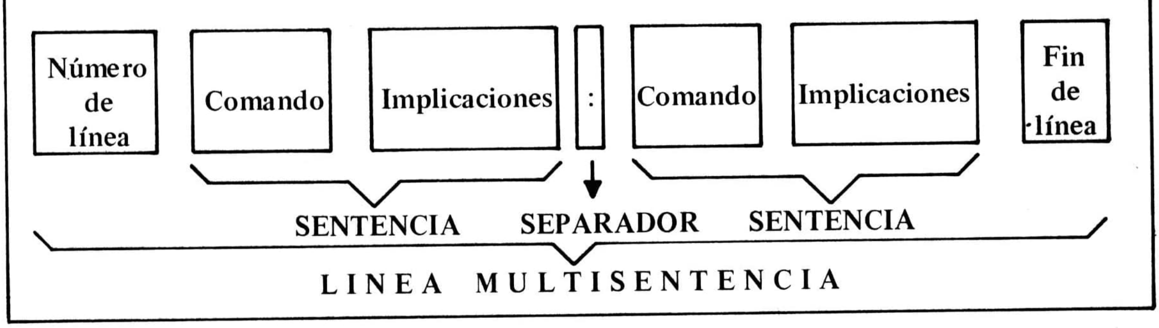
LINEAS DE PROGRAMA

Una **línea de programa** constituye una sentencia que está formada por **cuatro campos:**



También existen las líneas de programa que tienen varias sentencias: son **líneas multisentencia,** y responden al esquema anterior, pero separando una sentencia de otra mediante **separadores,** que suelen ser *los dos puntos* (:).

La estructura de una línea multisentencia es:



Vamos a estudiar la línea de programa explicando cada uno de sus *campos.*

Gracias al **primero,** se establece la secuencia de lectura de las líneas de programa, siendo, pues, imprescindible que los números de línea vayan de menor a mayor, justo en el orden que el programador desee que se ejecuten.

Esto no implica que los números sean consecutivos. Lo más conveniente es numerar las líneas de diez en diez –o incluso más– de forma que siempre quede la posibilidad de introducir nuevas líneas con números intermedios.

Los ejercicios de los próximos epígrafes aclararán suficientemente este punto.

El segundo y **el tercer campos** conforman el cuerpo de la sentencia y son el verdadero

objeto de este CURSO. No obstante, podemos anticipar que *el segundo campo* estará ocupado siempre por alguna de las palabras del vocabulario BASIC, y *el tercer campo* podrá estar ocupado por una **variable**, una **constante**, una **expresión matemática** o una **cadena de caracteres.**

Entendemos por cadena de caracteres cualquier combinación de letras, números y símbolos.

El cuarto campo está destinado a indicar al ordenador que la línea ha terminado y, por tanto, debe prepararse para recibir otra nueva. El comando destinado a este fin tiene diferentes denominaciones, pero las más usuales son **ENTER** y **RETURN.**

EJEMPLO:

```
10 INPUT A$
20 PRINT A$
```

Este sencillo programa está compuesto por dos sentencias:

- La primera está en la línea de programa número 10 y ordena al computador que espere la entrada (INPUT) de una *variable alfanumérica* denominada A$.
- La segunda línea tiene el número 20 y ordena al computador que imprima en pantalla el contenido de la variable A$.

Recuerde que después de escribir cada una de las líneas anteriores debe teclear ENTER, RETURN o cualquier otro comando que, en su ordenador, indique *el fin de línea.*

ALGORITMO

Esta palabra se deriva de MUSA AL-KHOWARIZMI, nombre de un matemático árabe que vivió en el siglo IX.

El diccionario define la palabra **algoritmo** como "método y notación en las distintas formas de cálculo."

Un *algoritmo* es, en definitiva, un conjunto de operaciones perfectamente especificadas cuyo objeto es obtener un resultado partiendo de unos datos.

EJEMPLO:

¿Cuál es el algoritmo que nos permite conocer la superficie de un cuadrado en función de la longitud de su lado?

SOLUCION:

S: superficie del cuadrado.

l: lado del cuadrado.

Algoritmo buscado: **S = l * l** (1)

En este caso, el dato necesario es la longitud del lado del cuadrado y el resultado que se obtendrá es la superficie de esta figura geométrica.

El conjunto de operaciones de este algoritmo se reduce a multiplicar el valor de **l** por sí mismo.

El algoritmo *S = l * l* es válido para cualquier valor de *l* y, en general, de las variables que intervienen.

(1) Utilizaremos el signo * para indicar la multiplicación, por ser el que figura usualmente en las computadoras.

CONSTANTES

Todos los valores que permanecen invariables en un algoritmo se llaman **constantes.**

EJEMPLO:

S: superficie del triángulo.
b: base del triángulo.
h: altura del triángulo.

$$S = \frac{b * h}{2}$$

En este algoritmo, que determina la superficie de un triángulo, tenemos *la constante* 1/2.

En Informática hay dos tipos de constantes: **alfanuméricas** y **numéricas.**

Las constantes alfanuméricas están formadas por cualquier combinación de caracteres, siempre que estén escritos entre comillas. Por ejemplo, "SEAT 127" ó "1, 2, 3... Responda otra vez".

Las constantes numéricas son números positivos o negativos, enteros o decimales, y no van entre comillas. La coma decimal se representa por un punto (.).

Existen dos tipos de *constantes numéricas:*

A. *Enteras*

Están formadas por cualquier número, positivo o negativo, y no pueden tener decimales. Por ejemplo, 21729 ó –18617.

B. *Reales*

Una constante real es cualquier elemento del conjunto de los números reales. Es decir, abarca todos los números positivos y negativos, enteros y decimales. Por ejemplo, 217, –3142, 2.147 ó –34.4157.

VARIABLES

Una **variable** representa un valor que *puede cambiar* a lo largo de un algoritmo, pero que *permanece fijo* mientras no haya una instrucción que indique lo contrario. *Las variables* también pueden ser **numéricas** y **alfanuméricas.**

En BASIC, *las variables numéricas* se representan por cualquier conjunto de caracteres, con la única condición de que el primero sea una letra. Por ejemplo, LET a = 5; LET A = 7; LET AB = 15, o LET AB1 = 45. (1).

Las variables alfanuméricas o *de caracteres* se representan por una sola letra, mayúscula o minúscula, seguida del símbolo $. De esta forma, el computador distingue que el contenido de esa variable es distinto de un número y lo interpretará como una cadena de caracteres. Esta cadena debe estar entrecomillada. Por ejemplo, LET a$ = "ordenador", o LET P$ = = "país".

Se entiende por **nombre de la variable** cualquier combinación de caracteres –de acuerdo con la sintaxis anteriormente explicada– que la represente. En los ejemplos anteriores, hemos denominado **a, A, AB** y **AB1** las variables numéricas, y **a$** y **P$** las variables alfanuméricas.

---

(1) LET es el comando o instrucción BASIC que asigna un valor a una variable, como se verá más adelante.

EXPRESIONES

Anteriormente nos hemos referido a las variables como elementos capaces de contener un valor. Una **expresión**, por el contrario, **es un valor.**

Nada impide, sin embargo, que el valor de una expresión varíe de acuerdo con los valores de las variables que forman parte de la expresión.

EJEMPLO:

Definir una expresión que determine el valor del importe de una conferencia a Sevilla, sabiendo que cada "paso" cuesta 10 ptas.

SOLUCION:

Importe de la conferencia = **10 * n.**

En esta expresión **10 * n** tenemos:

- Una constante numérica **10**, que es el precio de un "paso".
- Una variable numérica **n**, que representa el número de "pasos".

Evidentemente, para cada valor de n tendremos un valor diferente de la expresión, que nos dará los diferentes importes de las conferencias a Sevilla, según su número de pasos.

Las expresiones matemáticas se escriben en BASIC prácticamente igual que con lápiz y papel, pero con las variaciones impuestas por los símbolos que figuran en el teclado del ordenador. Tal es el caso, por ejemplo, de x^3 (equis elevado a 3), que nos obligará a teclear **x ↑ 3** ó **x ∧ 3**, según el tipo de ordenador que utilicemos.

El orden de prelación de las diferentes operaciones matemáticas se verá más tarde, pero, antes de llegar al mismo, es importante hacer observar unas sutiles diferencias que existen entre escribir una expresión matemática a mano o en un computador. Supongamos que deseamos transcribir la expresión $\frac{a + b}{2\,c}$ de forma entendible para el ordenador. Para lograrlo, no debemos olvidar ningún signo —como * entre 2 y **c** en el denominador— y no magnificar la capacidad de la máquina, ya que:

- Si escribimos **a + b/2 * c**, el ordenador interpretará $a + \frac{b}{2} * c$.
- Si escribimos **(a + b)/2 * c**, interpretará $\frac{a + b}{2} * c$.
- Sólo podrá interpretar correctamente la expresión citada si escribimos **(a + b)/(2 * c)**.

OPERADORES

Los operadores son símbolos que representan el conjunto de todas **las operaciones –aritméticas** o **lógicas**– que se pueden realizar entre valores.

Estos valores pueden ser –obviamente–, a su vez, expresiones. Son **los operandos.**

Como puede observarse en el ESQUEMA DE COMANDOS BASIC, se dispone de cuatro tipos de operadores: *aritméticos, de relación, lógicos* y *funcionales.*

OPERADORES ARITMETICOS

Se denominan así porque actúan entre valores aritméticos, siendo el conjunto de los símbolos que los componen y su orden de prioridad establecidos por el ordenador los siguientes:

PRIORIDAD	OPERACION	OPERADOR
máxima →	exponenciación	↑ (a veces ∧)
	multiplicación y	*
	división	/
	suma y	+
mínima ←	resta	–

Para operadores de la misma prioridad, la máquina ejecuta las operaciones de izquierda a derecha.

Las operaciones dentro de un paréntesis las efectúa primero y siguiendo el orden de prioridad anterior.

EJEMPLO:

(6 * 5 ↑ 3 / (4 + 2) / (3 – 1)

1° Calcula la potencia 5 ↑ 3 (5^3)
2° Multiplica el resultado por 6.
3° Calcula 4 + 2
4° Divide el producto (2°) entre la suma (3°)
5° Calcula 3 – 1
6° Divide el cociente (4°) entre la diferencia (6°).

Las cadenas de caracteres sólo pueden sumarse.

La forma de yuxtaponer una cadena de caracteres a otra es intercalar entre ellas el operador +.

EJEMPLO:

PROGRAMA	COMENTARIO
1Ø LET A$ = "ABC"	Con las líneas 1Ø y 2Ø asignamos valores a las variables de caracteres A$ y B$.
2Ø LET B$ = "DEF"	
3Ø PRINT A$ + B$	Con la línea 3Ø ordenamos la impresión del contenido actual de A$ y B$, justamente uno a continuación del otro.

El resultado de ejecutar este programa (¡recuerde: RUN!) será:

ABCDEF

OPERADORES DE RELACION

Estos operadores sólo actúan en proposiciones entre dos operandos, cuyo resultado únicamente puede ser **CIERTO**, que se representará por 1, o **FALSO**, que se representará por Ø.

Sus símbolos son:

igual que	=
distinto que	< >
menor que	<
mayor que	>
menor o igual que	< =
mayor o igual que	> =

Si un operador de relación compara dos expresiones en las que intervienen operadores aritméticos, antes de efectuar la comparación, actúan los operadores aritméticos.

EJEMPLO:

Determinar si 2 + 5 es menor que $(15 + 4)^3$ /20250

SOLUCION:

Es evidente que, sin efectuar las operaciones aritméticas que determinan el valor de cada expresión, es imposible responder a la proposición anterior.

Las cadenas de caracteres pueden ser comparadas con estos mismos operadores. Para hacerlo se cotejan, carácter a carácter, ambas cadenas valorando cada carácter de acuerdo con su **código ASCII**. Más adelante se estudiará todo lo referente a los códigos, baste por ahora saber que cada carácter tiene su propio valor.

OPERADORES LOGICOS

Responden directamente a las **funciones lógicas del álgebra de BOOLE.**

Los operadores lógicos actúan entre operandos en los que, a su vez, intervienen operadores de relación. Esto quiere decir que los operandos de los operadores lógicos tendrán siempre el valor **1** (CIERTO) ó **Ø** (FALSO), y, por consiguiente, *los operadores lógicos* sólo responden **1** ó **Ø** (CIERTO o FALSO).

Los operadores lógicos más usuales, según su orden de prioridad, son **NOT, AND** y **OR,** que se estudiarán con detalle más adelante. Hay computadores que tienen también **XOR, EQU** e **IMP.**

Ver apéndice FUNCIONES LOGICAS APLICADAS.

OPERADORES FUNCIONALES

Estos son algoritmos que residen en el propio BASIC y operan sobre datos suministrados al computador por medio del teclado o el programa, obteniendose como resultado un valor.

Las funciones de este tipo pueden ser numéricas, alfanuméricas y operativas.

En el ESQUEMA DE COMANDOS BASIC se puede ver el conjunto de estos operadores.

CONFIGURACION DE LOS CAPITULOS

En los capítulos siguientes se estudia el vocabulario BASIC más usual.
Cada capítulo responde a la siguiente estructura:

PALABRA	SINTAXIS DE LA PALABRA	TRADUCCION

INTERPRETACION	Explica el uso general de la palabra.
POSIBILIDADES	Explica las diferentes posibilidades de utilización.

FORMA DE TECLEAR LA INSTRUCCION

EJEMPLOS	Para aclarar el campo de utilización.
DIDACTICA	Sugiere el método de enseñanza/aprendizaje de la palabra.
EJERCICIOS	Sugiere y/o plantea ejercicios para la enseñanza/aprendizaje.

Antes de que inicie el estudio del primer capítulo, que le ayudará a introducirse en el lenguaje de programación BASIC, permítanos darle...

Un consejo: Si no tiene microordenador, procure proveerse de uno. Sin él tendrá dificultades para asimilar cuanto a continuación se expone.

Otro consejo: Siempre que piense qué podría suceder si hace tal o cual cosa, ¡NO LO DUDE Y HAGALA!, ésta es la mejor vía para aprender.

Y un consejo final: Provéase de un cuaderno y escriba en él todos los ejercicios que realice. Le será muy útil a la hora de confeccionar sus propios programas.

METODOLOGIA PARA LA INTRODUCCION A LOS COMANDOS

En la DIDACTICA de cada comando se sugiere el método de enseñanza/aprendizaje de la palabra mediante actividades de *recuerdo, explicación* y *observación experimental,* encaminadas a que los alumnos recorran las etapas lógicas de todo proceso de aprendizaje: **percibir, comprender, asimilar** y **utilizar** los conocimientos impartidos por el profesor.

Parece innecesario advertir que cada profesor puede utilizar la metodología que estime más oportuna para conseguirlo, por lo que el lector debe aceptar la que a continuación se expone como una mera sugerencia, ecaminada a conseguir un doble objetivo:

- Por una parte, involucrar a todos los alumnos en un proceso de interactividad total.
- Por otra, llevar al estudiante, indirecta y paulatinamente, a los modos y maneras que tiene de *razonar* un computador, concibiendo el comando BASIC más allá de su mera traducción del inglés al español para llegar a toda la estructura mental que envuelve.

El proceso metodológico que proponemos se desarrollará a través de las siguientes fases:

1. **Recordar** los conocimientos adquiridos anteriormente que se indican en el apartado de DIDACTICA de cada comando.

2. **Presentar** el comando que se va a estudiar mediante el ordenador.

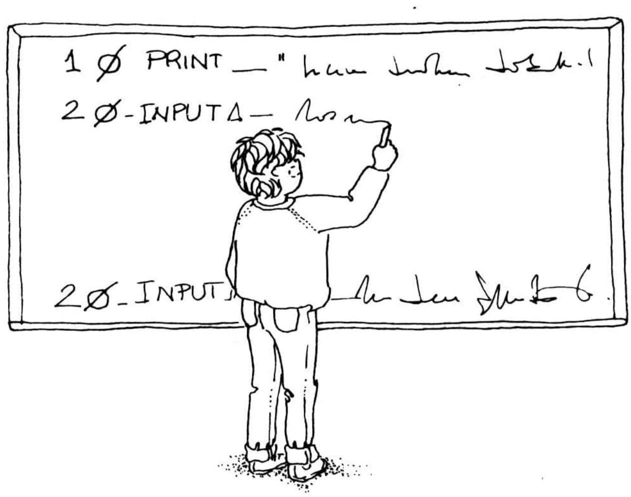
3. **Explicar** el comando y sus posibilidades de uso.
   En esta fase, debe utilizarse un mural en el que esté representado el teclado del ordenador que se utiliza en clase. Sobre este teclado, los alumnos simularán apretar las teclas en el momento oportuno.
   La pizarra de la clase servirá de pantalla sobre la que un alumno —que simulará la actuación del ordenador— irá escribiendo lo que se introduzca mediante el teclado del mural.
   Un ejemplo sencillo aclarará lo expuesto. El profesor, dirigiéndose a un alumno, le dará instrucciones de este tipo:

   1Ø PRINT "papá"

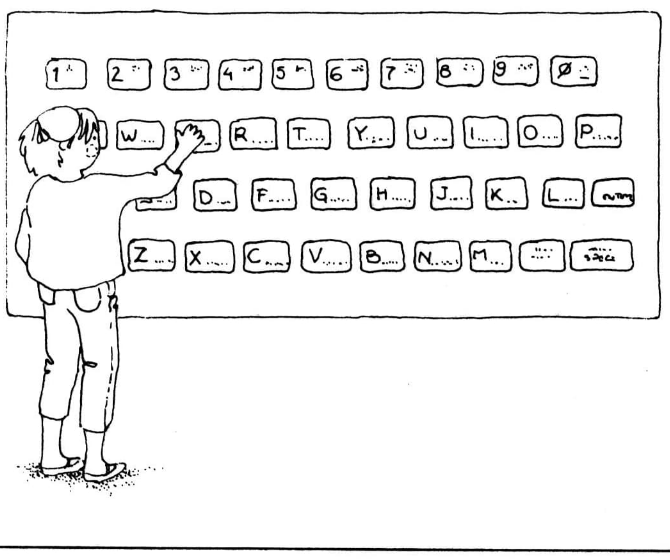
   El alumno designado irá tocando las teclas correspondientes y el alumno que simula la actuación del ordenador irá escribiendo lo que *teclee* su compañero al pie de la pizarra (pantalla imaginaria del monitor).
   Cuando el teclista u operador pulse la tecla **ENTER,** el alumno-ordenador escribirá la palabra **papá** a partir del ángulo superior izquierdo de la pizarra.
   Durante este proceso, los demás alumnos deben observarlo y señalar los posibles fallos cometidos por el teclista o el alumno-ordenador.
   En función de la edad y el nivel de los alumnos, el profesor desarrollará estas sugerencias metodológicas para el resto de las posibilidades del comando.



El profesor, dirigiéndose a un alumno, le dará las instrucciones que debe teclear.



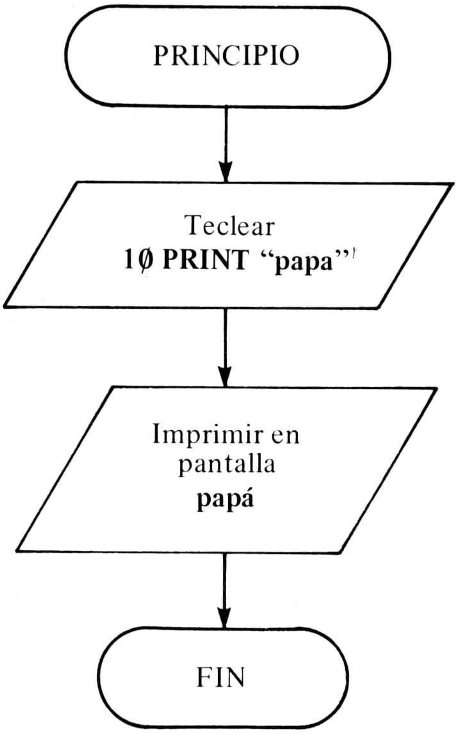
La pizarra de la clase servirá de pantalla sobre la que el alumno —que simulará la actuación del ordenador— irá escribiendo lo que se introduzca mediante el teclado del mural.



En el mural está representado el teclado del ordenador que se utiliza en clase. Sobre este teclado, los alumnos simularán apretar las teclas en el momento oportuno.

4. Una vez realizada esta fase, es aconsejable desarrollar un pequeño *diagrama de flujo* que establezca sólidamente el proceso lógico para, de esta forma, estudiar el comando en profundidad.
Un diagrama de flujo es la representación gráfica de un programa mediante símbolos convencionales.
Los diagramas de flujo [1] sirven para organizar la estructura de los programas; para conocer los pasos que sigue el ordenador en su ejecución, y para habituar a quien los utiliza a seguir un proceso lógico en la solución de los problemas.
En el caso del ejemplo propuesto, el diagrama podría ser el siguiente:

DIAGRAMA — COMENTARIOS



Se inicia el proceso.

El paralelogramo indica una entrada o salida.
Por el contenido de este programa, es una entrada por el teclado.

Es también una entrada o salida. En este caso, se trata de una salida a pantalla.

Termina el proceso.

Estos diagramas irán ganando en concisión a medida que los conceptos se consoliden y no sea necesario determinar en ellos los sucesivos pasos de forma tan exhaustiva. Mediante la utilización de *los diagramas de flujo* los alumnos llegan a comprender con mayor facilidad los comandos y su sintaxis, al tiempo que perfeccionan su pensamiento lógico y adquieren hábitos en la resolución de los problemas

5. **Asimilar** las posibilidades de utilización del comando con la observación experimental de las actividades que se proponen, mediante su realización individual utilizando las máquinas.

6. **Utilizar** el comando para realizar *los ejercicios* que se proponen y, así, consolidar y reforzar el aprendizaje.

---

(1) Ver el apéndice DIAGRAMAS DE FLUJO.

TECLEANDO UN PROGRAMA

Al estudiar la estructura de una línea de programa, vimos que el cuarto campo está dedicado a indicar al computador el FIN DE LINEA. Esto quiere decir que, una vez tecleada la línea completa, debemos apretar **ENTER** (o **RETURN**, según el modelo de computador) para ordenar a la máquina que la memorice, cosa que hará si no hay ningún error de sintaxis.

Supongamos que la línea que vamos a introducir es:

152Ø INPUT A$

En primer lugar, teclearemos el número de línea: **152Ø**; a continuación, el comando **INPUT**, y, finalmente, los caracteres **A** y **$**. Con todo esto la línea habrá concluido y sólo faltará indicar al computador que la pase a su memoria, para lo cual apretaremos la tecla **ENTER** (o **RETURN**, según se ha dicho).

Todos los computadores tienen un símbolo –normalmente >– para indicar la última línea memorizada, que se sitúa automáticamente entre el número de línea y la sentencia. Este indicador se llama **puntero.**

Si la línea no es aceptada por el ordenador debido a algún tipo de error, deberemos proceder a corregirlo.

En primer lugar, debemos observar que, en la zona de la pantalla donde van apareciendo los diferentes caracteres a medida que se teclean, un **cursor** se va desplazando y queda siempre a la derecha del último carácter tecleado.

Este *cursor* está representado por diferentes símbolos, según el tipo de ordenador que se esté empleando.

El cursor intermitente indica dónde está situado, y la forma en que queda representado en la pantalla no importa tanto como el hecho de saberlo desplazar, voluntariamente, a derecha e izquierda a lo largo de la línea que estemos manipulando.

Normalmente, en los teclados de las computadoras aparecen los símbolos ← y →, con los cuales se consigue este efecto.

En cuanto a la forma de utilizarlos deberá recurrirse a las indicaciones dadas por el MANUAL de la máquina en cuestión.

Visto esto, hay que desplazar *el cursor* justo a la derecha del carácter equivocado y borrarlo apretando la tecla DELETE o aquélla que el ordenador tenga para tal fin.

Supongamos que en la línea

1Ø INPUT A$

queremos sustituir **A** por **B**. Una vez situado *el cursor* a la derecha de **A**, hay que apretar la tecla DELETE –o la que tenga el ordenador, según dijimos más arriba–, con lo que se borrará **A** y sólo restará escribir **B** en su lugar.

Hay que destacar que no hay nada que impida, una vez posicionado *el cursor* en el lugar adecuado, insertar nuevos elementos en la línea.

Todos estos detalles se verán con más claridad a medida que avancemos en la lectura de esta publicación. Sin embargo, podemos anticipar que, si la línea de programa que queremos corregir está ya memorizada por el computador, podemos manipularla de acuerdo con las indicaciones que, para tal fin, especifica cada ordenador. Este proceso se llama **"editar la línea"**.

DIDACTICA:

* Antes de introducir a sus alumnos en el contenido de los capítulos siguientes, aquéllos deben familiarizarse con el microcomputador:
  - Qué sistemas físicos van a utilizar: televisor, grabadora de casetes, etc.
  - Cómo se conectan entre sí.
  - Cómo es el teclado del microcomputador y cómo se maneja. Los alumnos deben practicar hasta conseguir ciertos hábitos.

* Recuerde a los alumnos:
  - Qué es una *línea de programa* y qué *campos* la componen.
  - Qué es una *sentencia* y qué una *multisentencia.*
  - Para qué sirve cada uno de los campos de una línea de programa.
  - Los conceptos de *algoritmo, constante, variable, expresión, operador, operandos* y las clases de operadores: *aritméticos, de relación, lógicos* y *funcionales.* No olvide seleccionar y adaptar los conceptos explicados a la mentalidad de sus alumnos.
  - Qué son *el puntero* y *el cursor*, y para qué sirven.

EJERCICIOS:

1. Prácticas en la conexión de los sistemas físicos:
   - 1.1. La computadora
   - 1.2. El monitor o receptor de televisión.
   - 1.3. El magnetófono de casete, para reproducir y para grabar un programa.
   - 1.4. Otras máquinas: impresora, microdrive, etc.
2. Prácticas en el manejo del teclado:
   - 2.1. Mecanografía.
   - 2.2. Palabras clave.
   - 2.3. Modos del cursor, si existen en el ordenador de prácticas.
   - 2.4. Tecleado de programas sencillos.
   - 2.5. Corrección de líneas mal escritas.
   - 2.6. Edición y modificación de líneas ya memorizadas.

COMANDOS O INSTRUCCIONES BASIC FUNDAMENTALES

PRINT. PRINT expresiones. IMPRIMIR
LET. LET variable = expresión. DEJAR
INPUT. INPUT "texto"; variable. ENTRADA
REM. REM aclaraciones. REMEMORAR
STOP/CONT. STOP/CONT. PARAR/CONTINUAR
GO TO. GO TO número de línea. IR A
IF/THEN. IF expresión THEN sentencia. SI ... /ENTONCES ...
FOR ... TO/NEXT. PARA ... HASTA/PROXIMO
FOR ... TO ... STEP/NEXT. PARA ... HASTA ... SALTO/PROXIMO
GO SUB/RETURN. GO SUB número de línea. IR A SUBRUTINA/VOLVER
READ/DATA. LEER DATOS
  DATA. DATA constantes. DATOS
  READ. READ variables. LEER
RESTORE. RESTORE número de línea. VOLVER A ALMACENAR
INT. INT (expresión numérica). ENTERO
RND/RANDOMIZE. ALEATORIO/ALEATORIZAR
  RND. RND. ALEATORIO
  RANDOMIZE. RANDOMIZE expresión numérica. ALEATORIZAR

OTRAS FUNCIONES INCORPORADAS:

SQR. SQR (expresión numérica). RAIZ CUADRADA
SGN. SGN (expresión numérica). SIGNO
ABS. ABS (expresión numérica). ABSOLUTO
LN o LOG. LN (expresión numérica). LOGARITMO
EXP. EXP (expresión numérica). EXPONENCIAL
SIN. SIN (expresión numérica). SENO
COS. COS (expresión numérica). COSENO
DEF FN. DEF FN variable = expresión. DEFINIR FUNCION
FN. FN variable (valor de la variable). FUNCION

OPERADORES FUNCIONALES:

CHR$. CHR$ (expresión numérica). CARACTER
CODE o ASC. CODE (expresión alfanumérica). CODIGO
LEN. LEN (expresión alfanumérica). LONGITUD
STR$. STR$ (expresión numérica). CADENA
VAL. VAL (expresión alfanumérica). VALOR
INKEY$. INKEY$. APRETAR TECLA
DIM. DIM nombre de la matriz (definición de elementos). DIMENSIONAR

SENTENCIAS AUXILIARES:

LIST. LIST número de línea. LISTAR EN PANTALLA
LLIST. LLIST número de línea. LISTAR EN IMPRESORA
LOAD. LOAD "nombre". CARGAR
SAVE. SAVE "nombre". SALVAR
NEW. NUEVO

INTERPRETACION:

Significa *"imprimir"* y ordena la impresión en pantalla de la expresión o expresiones que figuran, optativamente, en las expresiones.

POSIBILIDADES:

* Si las expresiones son omitidas, una línea de pantalla salta en blanco.

** Si las expresiones son incluidas y estas expresiones son cadenas de caracteres, debemos escribirlas entrecomilladas.

*** Si las expresiones son incluidas y estas expresiones son matemáticas, aparecerán en pantalla sus valores. Recuerde que las expresiones matemáticas no se escriben entre comillas.

La posición de impresión de cada expresión de expresiones viene determinada por los símbolos que separan una expresión de otra:

- Un punto y coma (;) hace que la siguiente expresión se imprima justo a continuación de la inmediata anterior.
- Una coma (,) obliga a imprimir a partir de la siguiente mitad de la pantalla o al comienzo de la línea siguiente si la otra mitad ya se ocupó.
  Algunos dialectos BASIC dividen la línea en cuatro zonas, obligando con la coma a comenzar la impresión al cuarto de pantalla correspondiente.

Si las expresiones finalizan con una (,) o un punto y coma (;), la siguiente instrucción PRINT, que pudiera aparecer a lo largo del programa, comenzaría la impresión de sus expresiones conforme al criterio expuesto más arriba.

En caso de no acabar con una coma o un punto y coma, la siguiente instrucción PRINT comenzaría la impresión de sus expresiones en la siguiente línea de pantalla que esté libre.

FORMA DE TECLEAR LA INSTRUCCION

Teclear **PRINT**, seguido de **ENTER**, para dejar en blanco una línea en la pantalla.

Teclear **PRINT y la expresión entre comillas** ("), seguido de **ENTER**, para que aparezca en pantalla *la expresión alfanumérica.*

Teclear **PRINT y la expresión**, seguido de **ENTER**, para que aparezca en pantalla *la expresión numérica.*

EJEMPLO:

1. Escriba un programa que imprima en pantalla la cadena de caracteres: **"La loba lamía, lentamente, los lobeznos 3 veces'."**

SOLUCION:

PROGRAMA	COMENTARIOS
1Ø PRINT "La loba ";	La forma de teclear esta línea de programa, respetan- las normas dictadas por cada MANUAL, sería: • Teclear **1Ø** • Teclear **PRINT** • Teclear " (comillas) • Escribir el texto • Dar un espacio • Teclear " (comillas) • Teclear ; (punto y coma) • Apretar **ENTER**

> Una vez seguidas las instrucciones dadas en COMENTARIOS, la línea número 1Ø de nuestro primer programa estará en la memoria del ordenador. En este caso, tenemos un programa, compuesto por una sola línea, cuya misión es imprimir el texto indicado cada vez que se le ordene al computador.

De una forma similar iríamos tecleando las siguientes líneas del programa:

PROGRAMA	COMENTARIOS
2Ø PRINT "lamía, lentamente, ";	• Teclear **2Ø** • Teclear **PRINT** • Teclear " (comillas) • Escribir el texto • Dar un espacio • Teclear " (comillas) • Teclear ; (punto y coma) • Apretar la tecla **ENTER**
3Ø PRINT "los lobeznos ";	• Teclear **3Ø** • Teclear **PRINT** • Teclear " • Escribir el texto • Dar un espacio • Teclear " • Teclear ; • Apretar **ENTER**
4Ø PRINT "3 veces."	• Teclear **4Ø** • Teclear **PRINT** • Teclear • Escribir el texto • Teclear " • Apretar **ENTER**

Para ordenar al computador que EJECUTE el programa que tiene en memoria, el BASIC dispone del comando RUN.

Así, en el caso que nos ocupa, si tecleamos RUN y, a continuación, ENTER, inmediatamente aparecerá en la pantalla:

La loba lamía, lentamente, los lobeznos 3 veces.

Una vez en este punto, es conveniente observar que la definición del ejemplo no fue del todo correcta, ya que decíamos: "Escriba un programa que imprima en...", cuando realmente deberíamos haber dicho: "Escriba un programa que, una vez ejecutado, determine la impresión en pantalla de..."

Esta precisión tiene por objeto fijar las ideas sobre las diferentes expresiones y conceptos que estamos manejando. Pasemos ahora a otros ejemplos.

EJEMPLO:

2. Escriba un programa que, una vez ejecutado, imprima en la pantalla la palabra **"Juan"**, que deje la siguiente línea en blanco e imprima en la siguiente **"excelente"**.

SOLUCION:

PROGRAMA	COMENTARIOS
1∅ PRINT "Juan" **2∅ PRINT**	La única novedad de este programa de tres líneas se presenta en la 2∅, con el comando PRINT sin ninguna expresión a continuación. Al ser ejecutado el programa, dejará una línea en blanco entre el anterior PRINT y el siguiente.
3∅ PRINT "excelente"	Ejecute el programa con RUN y ENTER, como ya sabe.

EJEMPLO:

3. Escriba el programa que, una vez ejecutado, imprima al comienzo de una línea de pantalla **"MUY"** y, en la siguiente zona de esa línea, **"BIEN"**.

SOLUCION:

PROGRAMA	COMENTARIOS
1∅ PRINT "MUY", "BIEN"	La coma introducida entre las dos cadenas entrecomilladas hará que la segunda cadena comience su impresión al principio de la siguiente zona de la misma línea. Ejecute el programa.

EJEMPLO:

4. ¿Cómo se producirá la impresión en pantalla de las cadenas de caracteres que figuran en el siguiente programa una vez ejecutado?

SOLUCION:

PROGRAMA	COMENTARIOS
10 PRINT "JUNTO "; **20 PRINT "A TI"** **JUNTO A TI**	El punto y coma con que acabamos la primera instrucción PRINT obliga a la expresión PRINT siguiente a imprimirse justo a continuación de la anterior. Observe que se ha dejado deliberadamente un espacio al final de la primera cadena (línea 10).

EJEMPLO:

5. Escriba un programa que, una vez ejecutado, imprima en pantalla el resultado de sumar 2 y 2.

SOLUCION:

PROGRAMA	COMENTARIOS
10 PRINT 2 + 2	Como observará, la expresión que sigue al comando PRINT va sin comillas, ya que es una expresión matemática. Las comillas sólo se usan para indicar al ordenador qué deseamos que una expresión sea tratada como una cadena de caracteres. En el caso que nos ocupa, ordenamos al computador que imprima el resultado de la operación indicada.

EJEMPLO:

6. Escriba un programa que, una vez ejecutado, imprima en pantalla: *2 + 2 son 4.*

SOLUCIONES:

PROGRAMA	COMENTARIOS
10 PRINT "2 + 2 son "; 2 + 2 o también: **10 PRINT** "2 + 2 son "; **20 PRINT** 2 + 2	La primera expresión, por ser entrecomillada, será tratada como una cadena de caracteres. Recuerde que, entre el último carácter de la cadena y las comillas, debe dejar un espacio. Así evitará que al ejecutar el programa aparezca en pantalla 2 + *2son 4.* ¡Cuide la corrección en la escritura! El punto y coma que separa las dos expresiones PRINT obliga al ordenador a que imprima el resultado de la operación matemática justo a continuación del último carácter de la cadena anterior.

Para dejar la pantalla limpia de los caracteres impresos con anterioridad, la mayoría de los dialectos BASIC disponen del comando **CLS** y/o del comando **CLEAR**.

Ambos serán estudiados posteriormente, pero hasta ese momento, usted puede **limpiar** la pantalla tecleando **CLS** y **ENTER** o **CLEAR** y **ENTER**. Procure utilizar la primera posibilidad, si dispone de ella, ya que la segunda –como veremos más adelante– deja todas las variables a Ø.

DIDACTICA:

* Explique a los alumnos:
  - Que PRINT es un comando o instrucción del BASIC.
  - Que PRINT es una palabra inglesa, cuyo significado en español es *"imprimir"*.

* Que los alumnos observen experimentalmente:
  - Qué sucede cuando se omite una expresión después de PRINT.
  - Cómo se escriben expresiones alfanuméricas y numéricas.
  - Cómo se pueden imprimir dibujos esquemáticos.
  - Cómo se ejecuta o corre un programa.
  - Cómo se limpia la pantalla de un carácter impreso erróneamente.
  - Cómo se limpia la pantalla utilizando CLS y ENTER y/o CLEAR y ENTER.

EJERCICIOS:

1. Escribir expresiones alfanuméricas de una y varias sentencias hasta que dominen el uso de la coma (,); el punto y coma (;), y las comillas (" "):

1.1. Escriba el siguiente programa y ejecútelo:

```
1Ø PRINT "Soy Juan,"
2Ø PRINT "soy de Madrid"
3Ø PRINT "y tengo 12 años."
```

1.2. Sin borrar el programa anterior, continúe escribiendo:

```
4Ø PRINT "Soy Juan, ";
5Ø PRINT "soy de Madrid  ";
6Ø PRINT "y tengo 12 años."
```

Ejecute los dos programas anteriores y compárelos. Explique las diferencias que hay entre ellos.

1.3. Sin borrar los dos programas anteriores, continúe escribiendo:

```
7Ø PRINT "Soy Juan,  ",
8Ø PRINT "soy de Madrid  ",
9Ø PRINT "y tengo 12 años."
```

Ejecute los programas anteriores y compare los resultados. Explique las diferencias que existen entre los tres.

2. Escribir expresiones numéricas:

2.1. Escriba el siguiente programa y ejecútelo:

```
1Ø PRINT 3 + 2
2Ø PRINT 3 – 2
3Ø PRINT 3 * 2
4Ø PRINT 3 / 2
5Ø PRINT 3 ↑ 2
```

2.2. Escriba el siguiente programa:

```
1Ø PRINT 2 * 3 + 4
2Ø PRINT 2 * (3 + 4)
3Ø PRINT (3 + 4) * 2
```

Ejecute el programa anterior y compare los resultados obtenidos. Explique las diferencias existentes.

2.3. Escriba el siguiente programa:

```
1Ø PRINT 3 * (3 + 4 – 5) / 2
2Ø PRINT (3 * 3 + 4 – 5) /2
3Ø PRINT (3 * 3) + 4 – 5 / 2
4Ø PRINT (3 * 3) + (4 – 5) / 2
5Ø PRINT ((3 * 3) + (4 – 5)) / 2
```

Ejecute el programa anterior, compare los resultados obtenidos y explique las diferencias existentes.

3. Realizar dibujos esquemáticos:

3.1. Escriba el siguiente programa y ejecútelo:

```
1Ø PRINT "XXXXXX"
2Ø PRINT "X        X"
3Ø PRINT "X        X"
4Ø PRINT "X        X"
5Ø PRINT "X        X"
6Ø PRINT "XXXXXX"
```

3.2. Trace las diagonales del cuadrado anterior.

3.3. Borre las diagonales del cuadrado anterior y uno de los lados. Transfórmelo en un rectángulo de doble base que altura.

3.4. Dibuje un triángulo rectángulo de catetos iguales.

3.5. Dibuje libremente.
Es conveniente diseñar los dibujos en un papel milimetrado antes de imprimirlos en pantalla: evitará muchos errores y pérdidas de tiempo... ¡y de paciencia!

SOLUCIONES

3.2.

```
 1 REM Ficha "PRINT"  Ej.3.2"
10 PRINT "XXXXXX"
20 PRINT "XX   X"
30 PRINT "X X  X"
40 PRINT "X  X X"
50 PRINT "X   XX"
60 PRINT "XXXXXX"
```

3.3.

```
  1 REM Ficha " PRINT " Ej.3.3
 10 PRINT "XXXXXXXXXXXX"
 20 PRINT "X          X"
 30 PRINT "X          X"
 40 PRINT "X          X"
 50 PRINT "X          X"
 60 PRINT "XXXXXXXXXXXX"
```

3.4.

```
  1 REM  Ficha" PRINT " Ej.3.4
 10 PRINT "X"
 20 PRINT "XX"
 30 PRINT "X X"
 40 PRINT "X  X"
 50 PRINT "X   X"
 60 PRINT "X    X"
 70 PRINT "X     X"
 80 PRINT "X      X"
 90 PRINT "X       X"
100 PRINT "XXXXXXXXXX"
```

LET	LET variable = expresión	DEJAR

INTERPRETACION

Asigna a una variable el valor de una expresión.

POSIBILIDADES:

Recordemos previamente que una *variable* contiene un *valor* que puede cambiar a lo largo de un algoritmo, pero que permanece fijo mientras no haya una instrucción que así lo determine.

Existen dos clases de variables: *numéricas* y *de caracteres.*

Las variables numéricas se representan por cualquier conjunto de caracteres —a veces, por una sola letra seguida de un número— con la condición de que el primero sea una letra. Ejemplo: LET ABc1 = *45.*

Las variables de caracteres se representan por una sola letra seguida del símbolo $. De esta forma, el computador sabe que el contenido de la variable será un texto, que deberá ser introducido entre comillas. Ejemplo: LET A$ = "Cervantes".

Al referirnos a las variables, dijimos que son elementos capaces de contener un valor. *Una expresión,* por lo contrario, *es un valor.*

Nada impide que el valor de una expresión varíe de acuerdo con los valores de las variables que forman parte de la expresión.

FORMA DE TECLEAR LA INSTRUCCION

Teclear **LET, el nombre de la variable numérica, el signo igual (=) y la expresión numérica,** seguido de **ENTER.**

Teclear **LET, el nombre de la variable alfanumérica, el signo igual (=) y la expresión alfanumérica entre comillas ("),** seguido de **ENTER.**

Recordar:

- Que el nombre de la variable numérica puede ser una letra o una letra seguida de otros caracteres.
- Que el nombre de la variable alfanumérica siempre es una letra seguida del símbolo $.

EJEMPLO:

1. Definir una expresión que determine el valor del importe de una conferencia con Sevilla, sabiendo que el valor de cada "paso" de contador lo podemos averiguar en la Telefónica.

SOLUCION:

Empezaremos por fijar y nombrar las variables que intervienen:

Llamaremos **P** a la variable que contendrá el valor de un "paso".

Llamaremos **N** a la variable que contendrá el número de "pasos".

Y llamaremos **I** a la variable que contendrá el importe de la conferencia con Sevilla.

Hecho esto, tendremos que definir *la expresión* que determine el valor del importe, en función de los valores que contengan las variables implicadas:

I = P * N

Es evidente que el importe de la conferencia será igual al resultado de multiplicar el número de "pasos" por el valor de un "paso".

Supongamos que, después de observar en el contador el número de pasos y de habernos informado en la Telefónica del valor de un "paso", los valores de las variables **P** y **N** son, respectivamente, **10** y **25**. Es decir, un "paso" vale 10 ptas. y el número de "pasos" de la conferencia ha sido 25. En función de esto, la forma de asignar variables en BASIC, teniendo en cuenta el comando LET, será:

```
LET P = 10
LET N = 25
LET I = P * N
```

Como podemos ver, la expresión en una variable numérica puede ser tanto un número como una expresión matemática, la cual, finalmente, es un número.

EJEMPLO:

2. Asignar a una variable de caracteres la expresión:
   El importe de su conferencia es.

SOLUCION:

LET A$ = "El importe de su conferencia es "

De esta forma, el ordenador sabe que la variable A$ contiene el texto entrecomillado.

Observe el espacio que queda entre *es* y las comillas. Sirve para evitar que la cifra que representa el valor de la conferencia quede impresa junto a *es*.

EJEMPLO:

3. Escriba un programa que, una vez ejecutado, nos dé el importe de una conferencia con Sevilla de 25 "pasos", siendo el valor de cada paso 10 ptas.

SOLUCION:

PROGRAMA	COMENTARIOS
1Ø LET P = 1Ø 2Ø LET N = 25 3Ø LET I = P * N 4Ø LET A$ = "El importe de su conferencia es " 5Ø PRINT A$; I	Con las cuatro primeras líneas asignamos nombre y contenido a las variables. Con la última línea nos limitamos a ordenar la impresión del contenido actual de las variables A$ e I.

Al ejecutar (RUN/ENTER) el programa anterior, aparecerá en pantalla:

El importe de su conferencia es 250

DIDACTICA:

* Recuerde a los alumnos:
  - La función de las comillas (" "), la coma (,) y el punto y coma (;)
  - Cómo se representan las variables numéricas y las variables de caracteres ($).
  - Que, desde un punto de vista matemático, la palabra LET viene a decir: *Deja la variable tal igual al valor cual.*

* Explique a los alumnos:
  - Que el comando LET asigna a una variable el valor de una expresión.
  - Que las variables pueden ser numéricas y de caracteres.
  - Cómo escribir un programa en el que intervienen una, dos o más variables numéricas y de caracteres.

* Que los alumnos observen experimentalmente:
  - La diferencia entre escribir PRINT A = *valor* y PRINT "A".
  - Que las variables alfanuméricas solamente pueden sumarse entre sí.
  - Que con las variables numéricas pueden realizarse todo tipo de operaciones aritméticas.

EJERCICIOS:

1. Escribir programas para resolver problemas referidos al propio entorno, en los que intervengan las operaciones aritméticas:

   1.1. Costes:

   1.1.1. ¿Cuánto valen 5 Kg de melocotones a 115 ptas. el Kg?
   1.1.2. ¿Cuánto valen 5 Kg de melocotones, a 115 ptas. el Kg, y 7 Kg de melón, a 96 ptas. el Kg? Hallar el valor total.
   1.1.3. ¿Cuánto me devolverán en la frutería, si para pagar 5 Kg de melocotones, a 115 ptas. el Kg, y 7 Kg de melón, a 96 pts. en Kg, entrego 1500 Ptas?

   1.2. *Ahorro:*

   1.2.1. Si Juan tiene 2500 Ptas. y gasta 1875 ¿cuánto le queda?
   1.2.2. Si Juan tiene 2500 ptas y gasta 325 ptas en el cine y 1550 ptas en dos libros ¿cuánto dinero le queda? (*)
   1.2.3. A Juan le dan sus padres 1000 ptas cada semana y gasta 325 pts en el cine y 500 en libros. ¿Cuánto dinero ahorra en 15 semanas?

   1.3. *Areas:*

   1.3.1. Hallar el área de un cuadrado de 6 cm de lado.
   1.3.2. Hallar el área de un círculo inscrito en un cuadrado de 6 cm de lado (*).

(*) Todos los ejercicios señalados con asterisco están sin resolver. Intente buscar Ud. mismo la solución. No es difícil.

1.3.3. Hallar el área de un círculo circunscrito a un cuadrado de 6 cm de lado.

1.3.4. Hallar el área de la corona circular comprendida entre las circunferencias inscrita y circunscrita a un cuadrado de 6 cm de lado.

1.3.5. Hallar el área lateral y el área total de un cilindro de 5 cm de radio y 12 cm de altura.

1.1. *Volúmenes:*

1.4.1. Hallar el volumen de un cubo de 10 cm de arista.

1.4.2. Hallar el volumen de un cilindro inscrito en un cubo de 10 cm de arista.

1.4.3. Hallar el volumen de un cilindro circunscrito a un cubo de 10 cm de arista.

1.4.4. Hallar la diferencia de volumen entre los cilindros inscrito y circunscrito a un cubo de 10 cm de arista.

2. Escribir programas, en los que intervengan variables alfanuméricas, referidos a los contenidos de las diferentes materias de estudio:

2.1. *Lenguaje:*

2.1.1. Vocabulario:
Dadas las dependencias de una casa (vestíbulo, comedor, cocina, etc.), escribir el programa que, al ejecutarlo, imprima en pantalla: *Las dependencias de mi casa son: el vestíbulo, etc.*

2.1.2. Gramática:
Dadas las diferentes clases de palabras (artículo, nombre, adjetivo, etc.), escribir el programa que, ejecutado, imprima en pantalla: *Las clases de palabras que pueden formar el sintagma nominal son: el artículo, el nombre, etc.*

2.2. *Geografía e Historia:*

2.2.1. Geografía:
Dada una relación de ríos españoles, escribir el programa que, ejecutado, imprima en pantalla: *Los ríos españoles que desembocan en el Atlántico son: el Miño, el Duero, etc*

2.2.2. Historia:
Dada una relación de personajes españoles, escribir el programa que, ejecutado, imprima en pantalla: *Son personajes importantes en el descubrimiento y conquista de América: Colón, Pizarro, etc.*

2.3. *Ciencias Naturales:*

2.3.1. Dada una relación de las partes de una planta, escribir el programa que, ejecutado, imprima en pantalla: *Las partes del aparato vegetativo de una planta son: la raíz, el tallo, etc.*

2.3.2. Dada una relación de vertebrados, escribir el programa que, ejecutado, imprima en pantalla:

Son mamíferos...
Son aves...
Son reptiles...
Son peces...
Son anfibios...

SOLUCIONES

1.1.1.

```
 1 REM Ficha " LET " Ej.1.1.1
10 LET k=115: REM "Precio del kg."
20 LET m=5: REM "Numero de kg."
30 PRINT "Valen ";k*m
```

1.1.2
```
 1 REM Ficha " LET " Ej.1.1.2
10 LET p=115: REM "Precio del kg.de melocoton"
20 LET p1=96: REM "Precio del kg.de melon"
30 LET k=5: REM "Kilos de melocoton"
40 LET k1=7: REM "Kilos de melon"
50 PRINT "El valor total es ";p*k+p1*k1
```

1.1.3
```
 1 REM Ficha " LET " Ej.1.1.3
10 LET p=115: REM "Precio del kg.de melocoton"
20 LET p1=96: REM "Precio del kg.de melon"
30 LET k=5: REM "Kilos de melocoton"
40 LET k1=7: REM "Kilos de melon"
50 PRINT "Me devolveran ";-p*k+-p1*k1+1500;" pts."
```

1.2.1
```
 1 REM Ficha " LET " Ej.1.2.1
10 PRINT "Le queda ";2500-1875
```

1.2.3
```
 1 REM Ficha " LET  " Ej.1.2.3
10 PRINT "Ahorra ";15*1000-15*325-15*500
```

1.3.1.
```
 1 REM Ficha " LET " Ej.1.3.1
10 PRINT "El area es ";6*6;" cm2."
```

1.3.3.
```
 1 REM Ficha " LET " Ej.1.3.3
10 PRINT "El area es ";2*PI*SQR 2*6;" cm2."
```

1.3.4.
```
 1 REM Ficha " LET " Ej.1.3.4
10 PRINT "El area es: ";PI*2*6^2/2^2-PI*6^2/2^2;" cm2."
```

1.3.5.
```
 1 REM Ficha " LET " Ej.1.3.5
10 PRINT "El area lateral es ";2*PI*5/2*12;" cm2."
20 PRINT "El area total es ";2*PI*5/2*12+2*PI*5^2/2^2;"
   cm2."
```

1.4.1.
```
 1 REM Ficha " LET " Ej.1.4.1
10 PRINT "El volumen es ";10^3;" cm2."
```

1.4.2.
```
 1 REM Ficha " LET " Ej.1.4.2
10 PRINT "El volumen es ";PI*10^2/2^2*10;" cm2."
```

1.4.3.
```
 1 REM Ficha " LET " Ej.1.4.3
10 PRINT "El volumen es ";PI*2*10^2/2^2*10;" cm2."
```

1.4.4.
```
 1 REM Ficha:" LET " Ej.1.4.4
10 PRINT "El volumen es ";PI*2*10^2/2^2*10-PI*10^2/2^2
   *10;" cm2."
```

2.1.1.
```
 1 REM Ficha:" LET " Ej.2.1.1
10 LET A$="vestibulo, "
20 LET B$="cocina, "
30 LET C$="comedor, "
40 LET D$="salon, "
50 LET E$="dormitorios, "
```

```
 60 LET F$="cuarto de bano, "
 70 LET G$="servicio, "
 80 LET H$="trastero, "
 90 LET I$="jardin, "
100 PRINT "Las dependencias de mi casa son ";
110 PRINT A$+B$+C$+D$+E$+F$+G$+H$+I$
```

2.1.2.

```
  1 REM Ficha:" LET " Ej.2.1.2
 10 LET A$="articulo, "
 20 LET B$="nombre, "
 30 LET C$="adjetivo, "
 40 LET D$="pronombre, "
 50 LET E$="verbo, "
 60 LET F$="adverbio, "
 70 LET G$="preposicion, "
 80 LET H$="conjuncion, "
 90 PRINT "Las clasesde palabras que pueden formar el
    sintagma nominal, son ";
100 PRINT A$+B$+C$
```

2.2.1

```
  1 REM Ficha:" LET " Ej.2.2.1
 10 LET A$="Bidasoa, "
 20 LET B$="Nervion, "
 30 LET C$="Nalon, "
 40 LET D$="Navia, "
 50 LET E$="Mino, "
 60 LET F$="Duero, "
 70 LET G$="Tajo, "
 80 LET H$="Guadiana, "
 90 LET I$="GUadalquivir, "
110 LET K$="Ebro, "
120 LET L$="Ter, "
130 LET M$="Llobregat, "
140 PRINT "Los rios espanoles que  desembocan en el
   Atlantico, son ";
150 PRINT E$+F$+G$+H$+I$
```

2.2.2.

```
  1 REM Ficha:" LET " Ej.2.2.2
 10 LET A$="Colon, "
 20 LET B$="Hernan Cortes, "
 30 LET C$="Palafox, "
 40 LET D$="Nunez de Balboa, "
 50 LET E$="Viriato, "
 60 LET F$="Anibal, "
 70 LET G$="Francisco Pizarro, "
 80 LET H$="Elcano, "
 90 LET I$="Juan de Austria, "
100 PRINT "Son personajes importantes en eldescubrimie
    nto y la conquista deAmerica, ";
150 PRINT A$+B$+D$+G$+H$
```

2.3.1.
```
   1 REM Ficha:" LET " Ej.2.3.1
  10 LET A$="raiz, "
  20 LET B$="flor, "
  30 LET C$="tallo, "
  40 LET D$="fruto, "
  50 LET E$="pistilos, "
  60 LET F$="polen, "
  70 LET G$="corola, "
  80 LET H$="hojas, "
  90 LET I$="estambres, "
 100 PRINT "Las partes del aparato vegetativo de una
     planta, son ";
 150 PRINT A$+C$+H$
```

2.3.2.
```
   1 REM Ficha:" LET " Ej.2.3.2
  10 LET A$="conejo, "
  20 LET B$="tortuga, "
  30 LET C$="serpiente, "
  40 LET D$="perro, "
  50 LET E$="gato, "
  60 LET F$="gorrion, "
  70 LET G$="aguila, "
  80 LET H$="camello, "
  90 LET I$="salmon, "
 100 LET J$="sardina, "
 110 LET K$="jurel, "
 120 LET L$="ballena, "
 130 LET M$="rana, "
 140 PRINT "Son mamiferos,  ";
 150 PRINT A$+D$+E$+H$+L$
 160 PRINT "Son aves, ";
 170 PRINT F$+G$
 180 PRINT "Son reptiles, ";
 190 PRINT B$+C$
 200 PRINT "Son peces, ";
 210 PRINT I$+J$+K$
 220 PRINT "Son anfibios, ";
 230 PRINT B$+M$+B$
```

INPUT	INPUT "texto" ; variable	ENTRADA

INTERPRETACION

Produce una interrupción en la ejecución de un programa permitiendo, de este modo, la entrada de información a través del teclado.

POSIBILIDADES:

Cuando un programador introduce un comando INPUT, está provocando una parada en el programa para, de esta forma, dar un contenido a la variable.

Si el "texto" ha sido escrito, éste aparecerá en la pantalla durante la interrupción y hasta que el dato solicitado haya sido tecleado y se apriete la tecla ENTER.

Si el "texto" no existe, sólo el *prompt* (1) típico del ordenador aparecerá en pantalla, indicando que está a la espera de información. En este caso, el dato que se introduzca será asignado a la variable.

Esta variable puede ser numérica o de caracteres y, consiguientemente, el dato introducido tiene que estar en consonancia con el tipo de variable. Es decir, si la variable se ha considerado como numérica, no se aceptará una cadena de caracteres como entrada del INPUT.

FORMA DE TECLEAR LA INSTRUCCION

Teclear **INPUT, el texto entre comillas ("), punto y coma (;) y el nombre de la variable**, seguido de **ENTER**, para que aparezca **el texto** en pantalla hasta que se introduzca por teclado **el valor de la variable,** seguido de ENTER.

Teclear **INPUT y el nombre de la variable,** seguido de **ENTER,** para que la pantalla quede **en blanco** hasta que se introduzca por teclado **el valor de la variable**.

EJEMPLO:

1. Escriba un programa que, una vez ejecutado, nos pida el número de "pasos" consumidos en una conferencia telefónica y, a continuación, el valor de cada "paso", para obtener finalmente el importe de la conferencia.

SOLUCION:

PROGRAMA	COMENTARIOS
1Ø INPUT "¿Número de pasos? "; N	Con las dos primeras líneas provocaremos, al ejecutar el programa, dos interrupciones para asignar valores a las variables **N** y **P.**

(1) Con la expresión *prompt* se viene a indicar que el ordenador está preparado para hacer su trabajo y, para ello, hace aparecer en pantalla un símbolo, al que por extensión se llama así.

PROGRAMA	COMENTARIOS
2∅ INPUT "Valor de un paso? "; P	
3∅ PRINT "El importe de su conferencia es ";	Con la tercera línea, ordenamos la impresión del texto que figura tras el comando PRINT.
4∅ PRINT N * P	Con la cuarta línea, ordenamos la impresión del resultado de multiplicar el contenido actual de las variables N y P.

Cuando ejecutemos este programa, tecleando RUN y ENTER, en la pantalla aparecerá escrito **¿Número de pasos?** y se parará a la espera del contenido de la variable **N.**

Si suponemos que el número de pasos ha sido **25**, tecleamos este número y, a continuación, ENTER, para indicar al ordenador que hemos concluido, aparecerá escrito en la pantalla **¿Valor de un paso?** y volverá a pararse a la espera del contenido de la variable **P.**

Si asignamos a esta variable el valor **1∅, lo cual implica teclear 1∅** seguido de ENTER, aparecerá en la pantalla **El importe de su conferencia es 25∅.**

Observe que el resultado de **N * P** se imprime justo a continuación del texto de la línea 3∅, debido a que éste concluye en el programa con un punto y coma (;).

EJEMPLO:

2. Escriba un programa que, una vez ejecutado, le pida el nombre de un escritor famoso; a continuación, le pida el título de una de sus obras, y, finalmente, imprima en pantalla ambas cosas.

SOLUCION:

PROGRAMA	COMENTARIOS
1∅ INPUT "¿Nombre del escritor? ";e$** **2∅ INPUT "Título de una obra suya? "; t$ **3∅ PRINT e$** **4∅ PRINT t$**	En las líneas 1∅ y 2∅ hemos definido el nombre de dos variables de caracteres. En las líneas 3∅ y 4∅ hemos ordenado la impresión de las variables de caracteres definidas anteriormente, una debajo de la otra.

DIDACTICA:

* Recuerde a los alumnos:
  - Cómo deben numerar las líneas de 1∅ en 1∅, o más, para permitir la entrada de nuevas sentencias y, así, poder ampliar el programa.
  - Qué ocurre al situar los signos ortográficos (coma, punto y coma, etc.) dentro y fuera de las comillas.
* Explique a los alumnos:
  - Que el comando INPUT produce una interrupción al ejecutar el programa, después de la impresión en pantalla del texto, si es que lo tiene, que permite la entrada de una información, la variable, por medio del teclado.

– Que la variable introducida puede ser numérica o de caracteres.
– Que se pueden combinar los comandos INPUT y PRINT de modo que se establezca un "diálogo" entre el ordenador y la persona que lo esté manejando.

* Que los alumnos observen experimentalmente:
– Cómo se introducen las variables numéricas y las variables de caracteres.
– Cómo se procede a teclear las comillas (" ") y el punto y coma (;).
– Cómo hay que teclear los espacios para que, al ejecutar el programa, en los textos impresos no aparezcan palabras juntas.

EJERCICIOS:

1. Escribir programas para resolver problemas referidos al propio entorno en los que intervengan las operaciones aritméticas:

1.1. Costes:

1.1.1. Calcular el valor de N unidades de una mercancía sabiendo el precio P de una unidad.
1.1.2. Calcular el coste total de N unidades de una mercancía, al precio P por unidad y de M unidades de otra, al precio Q por unidad.
1.1.3. Calcular la cantidad a devolver cuando se pagan X ptas. por la compra de N unidades de una mercancía, al precio P por unidad, y de M unidades de otra, al precio Q por unidad. Hallar el valor de cada mercancía, el valor total de ambas y la cantidad a devolver.

1.2. Ahorro:

1.2.1. Calcular el dinero que le sobra a una persona si tiene X y gasta Y.
1.2.2. Calcular el dinero que le sobra a una persona si tiene X y gasta Y en diversiones y Z en libros (*).
1.2.3. Calcular el dinero que ahorra una persona en M meses, si cada mes gana S y gasta V en vivienda; X en alimentación; Y en vestuario, y Z en transportes y diversiones.

1.3. Areas:

1.3.1. Hallar áreas de polígonos y figuras circulares.
1.3.2. Hallar áreas laterales y totales de poliedros y cuerpos redondos.
1.3.3. Hallar la base, la altura, el lado o la apotema de un polígono conociendo su área y los otros datos (*).

1.4. Volúmenes:

1.4.1. Hallar volúmenes de poliedros y cuerpos redondos.
1.4.2. Hallar una dimensión de un poliedro o cuerpo redondo conocidos el volumen y las otras dimensiones (*).

1.5. Móviles:

1.5.1. Hallar la velocidad V de un móvil que recorre el espacio E en el tiempo T.
1.5.2. Hallar el espacio E que recorre o el tiempo T que tarda un móvil conociendo los otros datos (*).

2. Escribir programas en los que intervengan variables alfanuméricas referidos a las diferentes materias de estudio:

2.1. Lengua y Literatura:

2.1.1. Vocabulario:
Escribir el programa que, ejecutado, imprima en pantalla nombres, adjetivos y verbos relacionados con la casa (*).

2.1.2. Gramática:
Escribir el programa que, ejecutado, imprima en pantalla:

– las clases de palabras que pueden formar el sintagma nominal;
– qué clase constituye el núcleo del sintagma nominal;
– qué clases de determinantes hay en este sintagma (*).

2.1.3. Literatura:
Escribir el programa que, ejecutado, imprima en pantalla una relación de obras literarias clasificadas por autores (*).

2.2. Geografía e Historia:

2.2.1. Geografía:
Escribir el programa que, ejecutado, imprima en pantalla una relación de accidentes costeros clasificados por los mares donde están (*).

2.2.2. Historia:
Escribir el programa que, ejecutado, imprima en pantalla una relación de personajes históricos y sus principales hazañas (*).

2.3. Ciencias Naturales:

2.3.1. Escribir el programa que, ejecutado, imprima en pantalla una relación de órganos del cuerpo humano clasificados por aparatos (*).

2.3.2. Escribir el programa que, ejecutado, imprima en pantalla una relación de vertebrados clasificados (*).

3. Escribir programas en los que intervengan combinadas variables numéricas y de caracteres que, una vez ejecutados, sirvan para:

3.1. Calcular la edad aproximada de una persona sabiendo el año en que nació y el actual.

3.2. Calcular los días transcurridos desde el principio del curso escolar y los días que faltan para que termine (*).

3.3. Calcular las horas de clase a lo largo de todo el curso escolar (*).

4. Idear sus propios programas utilizando los comandos conocidos: PRINT, LET e INPUT.

NOTA: Es importante que conserve todos estos programas y los compare con los del capítulo siguiente.

SOLUCIONES

1.1.1.
```
  1 REM Ficha:" INPUT " Ej.1.1.1
 10 PRINT "Unidades: ";
 20 INPUT N;
 30 PRINT N
 40 PRINT "Precio Unidad: ";
 50 INPUT P
 60 PRINT P
 70 PRINT "TOTAL: ";N*P
```

1.1.2
```
  1 REM Ficha:" INPUT " Ej.1.1.2
 10 PRINT "Unidades A: ";
 20 INPUT N;
 30 PRINT N
 40 PRINT "Precio Unidad A: ";
 50 INPUT P
 60 PRINT P
 70 PRINT "COSTE UDS. A: ";N*P
 80 PRINT "Unidades B: ";
 90 INPUT N1;
100 PRINT N1
```

```
110 PRINT "Precio Unidad B: ";
120 INPUT P1
130 PRINT P1
140 PRINT "COSTE UDS. B: ";N1*P1
150 PRINT
160 PRINT "COSTE TOTAL: ";N*P+N1*P1
```

1.1.3.

```
  1 REM Ficha:" INPUT " Ej.1.1.3
 10 PRINT "Importe pagado: ";
 20 INPUT E
 30 PRINT E
 40 PRINT "Unidades A: ";
 50 INPUT N;
 60 PRINT N
 70 PRINT "Precio Unidad A: ";
 80 INPUT P
 90 PRINT P
100 PRINT "COSTE UDS. A: ";N*P
110 PRINT "Unidades B: ";
120 INPUT N1;
130 PRINT N1
140 PRINT "Precio Unidad B: ";
150 INPUT P1
160 PRINT P1
170 PRINT "COSTE UDS. B: ";N1*P1
180 PRINT
190 PRINT "COSTE TOTAL: ";N*P+N1*P1
200 PRINT "Cantidad a devolver: ";
210 PRINT -N*P+-N1*P1+E
```

1.2.1.

```
  1 REM Ficha:" INPUT " Ej.1.2.1
 10 PRINT "Dinero que tiene: ";
 20 INPUT X
 30 PRINT X
 40 PRINT "Gastado: ";
 50 INPUT Y
 60 PRINT Y
 70 PRINT "Dinero que le sobra: ";X-Y
```

1.2.3.

```
  1 REM Ficha:" INPUT " Ej.1.2.3
 10 PRINT "Dinero que gana al mes: ";
 20 INPUT S
 30 PRINT S
 40 PRINT "Gastado en vivienda: ";
 50 INPUT V
 60 PRINT V
 70 PRINT "Gastado en alimentacion: ";
 80 INPUT X
 90 PRINT X
100 PRINT "Gastado en vestuario: ";
110 INPUT Y
120 PRINT Y
```

```
130 PRINT "Gastado en Transportes y diversiones: ";
140 INPUT Z
150 PRINT Z
160 PRINT "Numero de meses: ";
170 INPUT M
180 PRINT M
190 PRINT "AHORRO: ";-V*M-X*M-Y*M-Z*M+S*M
```

1.3.1.

```
  1 REM Ficha:" INPUT " Ej.1.3.1
  2 PRINT "AREA DE POLIGONO REGULAR"'"Descomponer en
    triangulos"
 10 PRINT "Longitud del lado: ";
 20 INPUT L
 30 PRINT L
 40 PRINT "Longitud de la apotema: ";
 50 INPUT A
 60 PRINT A
 70 PRINT "Numero de lados: ";
 80 INPUT N
 90 PRINT N
100 PRINT "AREA: ";L*A/2*N
```

```
  1 REM Ficha:" INPUT " Ej.1.3.1
  2 PRINT "AREA DE TRIANGULOS"
 10 PRINT "Longitud de la base: ";
 20 INPUT L
 30 PRINT L
 40 PRINT "Longitud de la altura: ";
 50 INPUT A
 60 PRINT A
 70 PRINT "AREA: ";L*A/2
```

```
  1 REM Ficha:" INPUT " Ej.1.3.1
  2 PRINT "AREA DE PIRAMIDES REGULARES"
 10 PRINT "Longitud de un lado de la base: ";
 20 INPUT L
 30 PRINT L
 40 PRINT "Numero de lados: ";
 50 INPUT N
 60 PRINT N
 70 PRINT "Longitud de la apotema: ";
 80 INPUT A
 90 PRINT A
100 PRINT "AREA DE PIRAMIDE REGULAR: ";L*N*A/2
```

1.3.2.

```
  1 REM Ficha:" INPUT " EJ.1.3.1
  2 PRINT "AREA DE CILINDROS"
 10 PRINT "Longitud del radio de la base: ";
 20 INPUT R
 30 PRINT R
 40 PRINT "Longitud de altura: ";
 50 INPUT A
```

```
60 PRINT A
70 PRINT "AREA LATERAL: ";2*PI*R*A
```

```
 1 REM Ficha:" INPUT " EJ.1.3.2
 2 PRINT "AREA DE CILINDROS"
10 PRINT "Longitud del radio de la base: ";
20 INPUT R
30 PRINT R
40 PRINT "Longitud de la altura: ";
50 INPUT A
60 PRINT A
70 PRINT "AREA TOTAL: ";2*PI*R*A+2*PI*R^2
```

1.4.1.

```
  1 REM Ficha:" INPUT " Ej.1.4.1"
  2 PRINT "VOLUMEN DEL PARALELEPIPEDO"
 10 PRINT "Longitud del largo  de la base: ";
 20 INPUT L
 30 PRINT L
 40 PRINT "Longiud del alto de la base: ";
 50 INPUT A
 60 PRINT A
 70 PRINT "Altura de la figura: ";
 80 INPUT A1
 90 PRINT A1
100 PRINT "VOLUMEN: "
110 PRINT
120 PRINT ,L*A*A1
```

```
 1 REM Ficha:" INPUT " Ej.1.4.1"
10 PRINT "Longitud del diametro: ";
20 INPUT D
30 PRINT D
40 PRINT "SUPERFICIE DE LA ESFERA: ";PI*D^2
```

```
 1 REM Ficha:" INPUT " Ej.1.4.1"
10 PRINT "Longitud del diametro: ";
20 INPUT D
30 PRINT D
40 PRINT "VOLUMEN DE LA ESFERA: ";PI/6*D^3
```

1.5.1.

```
 1 REM Ficha:" INPUT " Ej.1.5.1"
10 PRINT "VELOCIDAD DE UN MOVIL"
20 PRINT "Espacio recorrido : ";
30 INPUT E
40 PRINT E
50 PRINT "Tiempo empleado: ";
60 INPUT T
70 PRINT T
80 PRINT "VELOCIDAD: ";E/T
```

3.1.

```
 1 REM Ficha :" INPUT " Ej.3.1"
10 PRINT "EDAD APROXIMADA DE UNA PERSONA"
```

```
20 PRINT "Ano de nacimiento: ";
30 INPUT A
40 PRINT A
50 PRINT "Ano actual: ";
60 INPUT A1
70 PRINT A1
80 PRINT "Edad :";A1-A;" anos"
```

INTERPRETACION:

Significa *rememorar, recordar* y permite introducir aclaraciones y observaciones que pueden ser de interés en una posterior revisión o lectura del listado del programa.

La sentencia REM no implica ningún tipo de proceso.

POSIBILIDADES:

Como se desprende de la nomenclatura utilizada –aclaraciones–, el texto que se utilice para las aclaraciones y observaciones es opcional. Esto quiere decir que si tecleamos REM en una línea de programa –bien a su comienzo, bien tras el separador (:)–, ésta quedará impresa para servir de indicador nemotécnico. En este sentido, es frecuente utilizar el separador (:) como medio de fragmentar un listado en "trozos" físicos para distinguir de un vistazo los diferentes elementos de un programa.

En todo caso, la sentencia REM se mantiene en el listado, tal como se ha escrito, sin ser procesada, continuando la ejecución del programa en la primera sentencia que la siga, bien tras un separador (:), bien en la siguiente línea del programa.

FORMA DE TECLEAR LA INSTRUCCION

Teclear **REM y las aclaraciones**, seguido de **ENTER.**

EJEMPLO:

Diferentes formas de introducir aclaraciones y observaciones en un listado:

PROGRAMA	COMENTARIO
1∅ REM Comienzo del programa.	REM con aclaraciones.
2∅ INPUT A	
3∅ REM	REM en solitario.
4∅ INPUT B	
5∅ :	Separadores para fragmentar el listado.
6∅ REM Suma	REM con aclaraciones
7∅ LET C = A + B: REM contador	REM tras un separador.
8∅ REM Fin: STOP	REM antes de un separador.
9∅ :	Separadores en forma de línea.

INTERPRETACION:

Cuando la lectura del programa llega a la instrucción **STOP**, finaliza la ejecución del mismo, pasando el computador a la situación de disponible.

Si el programa tiene más instrucciones después de STOP, para proseguir su ejecución es necesario teclear **CONT** seguido de ENTER.

POSIBILIDADES:

Cuando un programa detiene su ejecución, como consecuencia de un STOP, siempre existe la posibilidad de continuarla si se teclea –como se ha dicho más arriba–, antes que cualquier otra cosa, el comando CONT seguido de ENTER.

FORMAS DE TECLEAR LAS INSTRUCCIONES

Teclear **STOP**, seguido de **ENTER**, para *detener* la ejecución del programa.

Teclear **CONT**, seguido de **ENTER**, para *continuar* la ejecución del programa detenido por un STOP.

EJEMPLO:

PROGRAMA	COMENTARIOS
1Ø INPUT A **2Ø PRINT A** **3Ø INPUT B** **4Ø PRINT B** **5Ø PRINT A + B** **6Ø STOP** **7Ø PRINT (A + B) * 2** **8Ø STOP**	Una vez introducidos por el teclado los dos valores (números) que se solicitan en las variables de las líneas 1Ø y 3Ø, el programa detendrá su ejecución en la línea 6Ø imprimiendo el resultado de la suma. Si tecleamos CONT/ENTER, seguirá la ejecución del programa imprimiendo el resultado de la operación de la línea 5Ø y se detendrá en la línea 8Ø, donde hay otro STOP.

DIDACTICA:

* Recuerde a los alumnos:
  - Las posibilidades de utilización de los comandos PRINT, LET e INPUT.
  - Cómo se utiliza el comando LIST para listar en la pantalla el programa que el ordenador tiene en su memoria.
  - La conveniencia de utilizar el separador (:) para fragmentar un programa en sus diferentes partes.

* Explique a los alumnos:
  - Que el comando REM no afecta el programa, ya que el ordenador no lo tiene en cuenta en la ejecución.
  - Que el comando REM sirve para introducir aclaraciones en un programa.
  - Que las sentencias REM de un programa sólo pueden verse al hacer un listado del mismo en la pantalla.
  - Que el comando STOP sirve para finalizar la ejecución de un programa o de una parte del mismo.
  - Que el comando CONT sirve para proseguir la ejecución de un programa detenido por un STOP.

* Que los alumnos observen experimentalmente:
  - Cómo se lista un programa que el ordenador tiene en su memoria.
  - La utilidad de la sentencia REM para ver cómo está hecho un programa.
  - Cómo utilizar el separador (:) en un programa.
  - Cómo la sentencia STOP detiene la ejecución de un programa y cómo puede proseguirse pulsando CONT y ENTER.

EJERCICIOS:

Practicar sobre los programas realizados anteriormente:

1. Añadir aclaraciones y observaciones mediante el comando REM.
2. Utilizar separadores para fragmentar los listados.
3. Añadir sentencias STOP para separar las partes de un programa. Por ejemplo, en aquéllos que utilizan sucesivamente varias operaciones combinadas: costes, ahorro, áreas, volúmenes, etc.

INTERPRETACION:

Se ordena un salto incondicional a un número de línea.

POSIBILIDADES:

La secuencia de lectura de un programa, por parte del ordenador, sigue un orden creciente, yendo del número de línea más pequeño al mayor, de una forma sucesiva. Pero, cuando a lo largo de esta lectura, se encuentra con la orden GO TO, inmediatamente, el orden normal de la lectura se transfiere a la línea cuyo número está indicado en el número de línea.

FORMA DE TECLEAR LA INSTRUCCION

Teclear **GO TO y el número de línea**, seguido de **ENTER**, para ordenar que la secuencia de lectura del programa *siga en la línea del número dado.*

EJEMPLO:

1. Hagamos utilizable indefinidamente el programa que determinaba el importe de una conferencia.

SOLUCION:

PROGRAMA	COMENTARIOS
1Ø INPUT "¿Número de pasos? ";N **2Ø INPUT "¿Valor de un paso? ";P** **3Ø PRINT "El importe de su conferencia es ";N * P** **4Ø GO TO 1Ø**	Al añadir la línea 4Ø, obligamos seguir la lectura del programa a partir de la línea 1Ø tantas veces como queramos.

DIDACTICA:

* Explique a los alumnos:
  - Que el comando GO TO número de línea, produce un salto incondicional en la ejecución del programa a la línea cuyo número hemos tecleado. Es decir, altera el orden en la secuencia de lectura de un programa.
  - Que este salto puede ser hacia una línea anterior o posterior que aquélla donde está el comando GO TO. Es decir, a una línea de mayor o menor número que la de GO TO.

– Que cada vez que el orden vuelve a la lectura de una línea de programa anterior a la de GO TO, ya tiene en memoria los valores que han tomado las variables, y, en sentido contrario, las variables situadas en líneas de programa posteriores a un GO TO no son memorizadas hasta que la secuencia de lectura no obligue a pasar por esas líneas de programa.

– Que se puede emplear el comando GO TO para hacer utilizable indefinidamente un programa.

* Que los alumnos observen experimentalmente:

– Cómo una o más sentencias PRINT seguidas de un GO TO, que remite a la primera, se repiten indefinidamente.

– Cómo se repiten dichas sentencias PRINT en dos columnas cuando después de las comillas se teclea coma (" " ,).

– Cómo se repiten dichas sentencias PRINT a lo largo de todas las líneas cuando después de las comillas se teclea punto y coma (" ";).

– Cómo, mediante el comando GO TO, se pueden ordenar las sentencias de un programa puestas en desorden.

– Cómo la expresión **x = x + 1**, por ejemplo, que en Matemáticas no es correcta, sí es posible en Informática, ya que el ordenador calcula primero el valor situado a la derecha del signo (x + 1) y asigna ese valor a la variable situada a la izquierda del mismo **(x)**.

EJERCICIOS:

Utilizando los comandos conocidos (PRINT, LET, INPUT y GO TO):

1. Escribir programas para resolver problemas referidos al propio entorno en los que intervengan las operaciones aritméticas:

   1.1. Numeración:
   Contar y descontar de 2 en 2, de 3 en 3, etc., desde un número dado.

   1.2. Calculo:
   Confeccionar tablas de sumar, restar, multiplicar y dividir.

   1.3. Costes:
   1.3.1. Calcular el valor de N unidades de una mercancía sabiendo el precio P de una unidad.
   1.3.2. Calcular el coste total de N unidades de una mercancía, al precio P por unidad y de M unidades de otra, al precio 0 por unidad.
   1.3.3. Calcular la cantidad a devolver cuando se pagan X ptas. por la compra de N unidades de una mercancía, al precio P por unidad, y de M unidades de otra, al precio Q por unidad. Hallar el valor de cada mercancía, el valor total de ambas y la cantidad a devolver.

   1.4. Ahorro:
   1.4.1. Calcular el dinero que le sobra a una persona si tiene X y gasta Y.
   1.4.2. Calcular el dinero que le sobra a una persona si tiene X y gasta Y en diversiones y Z en libros (*).
   1.4.3. Calcular el dinero que ahorra una persona en M meses, si cada mes gana S y gasta V en vivienda; X en alimentación; Y en vestuario, y Z en transportes y diversiones

   1.5. Regla de interés:
   1.5.1. Hallar el interés I producido por un capital C, colocado al rédito R por ciento, durante el tiempo T.
   1.5.2. Hallar el capital C, el rédito R ó el tiempo T, conocidos los otros datos (*).

1.6. Regla de descuento:

Problemas similares a los anteriores (*).

1.7. Areas:

1.7.1. Hallar áreas de polígonos y figuras circulares.
1.7.2. Hallar áreas laterales y totales de poliedros y cuerpos redondos.
1.7.3. Hallar la base, la altura, el lado o la apotema de un polígono conociendo su área y otros datos (*).

1.8. Volúmenes:

1.8.1. Hallar volúmenes de poliedros y cuerpos redondos.
1.8.2. Hallar una dimensión de un poliedro o cuerpo redondo conocidos el volumen y las otras dimensiones (*).

1.9. Móviles:

1.9.1 Hallar la velocidad V de un móvil que recorre el espacio E en el tiempo T.
1.9.2. Hallar el espacio E que recorre o el tiempo T que tarda un móvil conociendo los otros datos (*).

2. Escribir programas en los que intervengan combinadas variables numéricas y de caracteres:

2.1. Calcular la edad de una persona sabiendo el año en que nació y el año actual.

2.2. Calcular los días transcurridos desde el comienzo del curso escolar y los días que faltan para que termine (*).

2.3. Calcular las horas de clase a lo largo de todo el curso escolar (*).

3. Idear sus propios programas utilizando los comandos conocidos: PRINT, LET, INPUT y GO TO.

NOTA: Recuerde que debe comparar estos programas con los del capítulo anterior. Podrá observar la ventaja de introducir el comando GO TO.

SOLUCIONES

1.1.

```
  1 REM Ficha:" GO TO "  Ej.1.1"
 10 PRINT "CONTAR DE N EN N"
 20 PRINT "Comienzo: ";
 30 INPUT C
 40 PRINT C
 50 PRINT "Numero a sumar: ";
 60 INPUT N
 70 PRINT N
 80 PRINT C+N
 90 LET C=C+N
100 GO TO 80
```

1.2.

```
  1 REM Ficha:" GO TO " Ej.1.2"
 10 PRINT "TABLA DE MULTIPLICAR"
 20 LET A=1
 30 PRINT "Numero: "
 40 INPUT N
 50 PRINT N;" por ";
 60 PRINT A;" = ";A*N
 70 LET A=A+1
 80 GO TO 50
```

1.3.1.

```
  1 REM Ficha:" GO TO " Ej.1.3.1"
  5 PRINT "UNIDADES POR PRECIO"
 10 PRINT "Unidades: ";
 20 INPUT N;
 30 PRINT N
 40 PRINT "Precio Unidad: ";
 50 INPUT P
 60 PRINT P
 70 PRINT "TOTAL: ";N*P
 80 PRINT
 90 GO TO 10
```

1.3.2.

```
  1 REM ficha:" GO TO " Ej.1.3.2"
  5 PRINT "CONJUNTO DE UNIDADES POR PRECIO"
 10 PRINT "Unidades A: ";
 20 INPUT N;
 30 PRINT N
 40 PRINT "Precio Unidad A: ";
 50 INPUT P
 60 PRINT P
 70 PRINT "COSTE UNIDADES A: "; N*P
 80 PRINT "Unidades B: ";
 90 INPUT N1;
100 PRINT N1
110 PRINT "Precio Unidad B: ";
120 INPUT P1
130 PRINT P1
140 PRINT "COSTE UNIDADES B: "; N1*P1
150 PRINT
160 PRINT "COSTE TOTAL: ";N*P+N1*P1
170 PRINT
180 GO TO 10
```

1.3.3.

```
  1 REM Ficha:" GO TO " Ej.1.3.3"
  5 PRINT "CALCULO DE SALDOS DE COMPRAS"
 10 PRINT "Importe pagado: ";
 20 INPUT E
 30 PRINT E
 40 PRINT "Unidades A: ";
 50 INPUT N;
 60 PRINT N
 70 PRINT "Precio Unidad A: ";
 80 INPUT P
 90 PRINT P
100 PRINT "COSTE UNIDADES A: "; N*P
110 PRINT "Unidades B: ";
120 INPUT N1;
130 PRINT N1
140 PRINT "Precio Unidad B: ";
150 INPUT P1
160 PRINT P1
170 PRINT "COSTE UNIDADES B: ";N1*P1
```

```
180 PRINT
190 PRINT "COSTE TOTAL: ";N*P+N1*P1
200 PRINT "Cantidad a devolver: ";
210 PRINT -N*P+-N1*P1+E
220 PRINT
230 GO TO 10
```

1.4.1.

```
  1 REM Ficha:" GO TO " Ej.1.4.1"
 10 PRINT "Dinero que tiene: ";
 20 INPUT X
 30 PRINT X
 40 PRINT "Gastado: ";
 50 INPUT Y
 60 PRINT Y
 70 PRINT "Dinero que le sobra: ";X-Y
 80 GO TO 10
```

1.4.3.

```
  1 REM Ficha:" GO TO " Ej.1.4.3"
 10 PRINT "Dinero que gana al mes: ";
 20 INPUT S
 30 PRINT S
 40 PRINT "Gastado en vivienda: ";
 50 INPUT V
 60 PRINT V
 70 PRINT "Gastado en alimentacion: ";
 80 INPUT X
 90 PRINT X
100 PRINT "Gastado en vestuario: ";
110 INPUT Y
120 PRINT Y
130 PRINT "Gastado en Transportes y diversiones: ";
140 INPUT Z
150 PRINT Z
160 PRINT "Numero de meses: ";
170 INPUT M
180 PRINT M
190 PRINT "AHORRO: ";-V*M-X*M-Y*M-Z*M+S*M
200 PRINT
210 GO TO 10
```

1.5.1.

```
  1 REM Ficha:" GO TO " Ej.1.5.1"
 10 PRINT "REGLA DE INTERES"
 20 PRINT "Capital: ";
 30 INPUT C
 40 PRINT C
 50 PRINT "% anual: ";
 60 INPUT R
 70 PRINT R
 80 PRINT "Tiempo en dias: ";
 90 INPUT T
100 PRINT T
110 PRINT "Interes: ";C*R*T/36000
```

```
120 PRINT
130 GO TO 20
```

1.7.1.

```
  1 REM Ficha:" GO TO " Ej.1.7.1"
  2 PRINT "AREA DE POLIGONO REG. INSCRITO"
  5 PRINT "Descomponer en triangulos"
 10 PRINT "Longitud del lado: ";
 20 INPUT L
 30 PRINT L
 40 PRINT "Longitud de la apotema: ";
 50 INPUT A
 60 PRINT A
 70 PRINT "Numero de lados: ";
 80 INPUT N
 90 PRINT N
100 PRINT "AREA: ";L*A/2*N
110 PRINT
120 GO TO 10
```

```
  1 REM Ficha:" GO TO " Ej.1.7.1"
  2 PRINT "AREA DE TRIANGULOS"
 10 PRINT "Longitud de la base: ";
 20 INPUT L
 30 PRINT L
 40 PRINT "Longitud de la altura: ";
 50 INPUT A
 60 PRINT A
 70 PRINT "AREA: ";L*A/2
 80 PRINT
 90 GO TO 10
```

1.7.2.

```
  1 REM Ficha:" GO TO " Ej.1.7.2"
  2 PRINT "VOLUMEN DE PIRAMIDES"
 10 PRINT "Area de la base: ";
 20 INPUT S
 30 PRINT S
 40 PRINT "Altura: ";
 50 INPUT A
 60 PRINT A
 70 PRINT "VOLUMEN: ";S*A/3
 80 PRINT
 90 GO TO 10
```

```
  1 REM Ficha:" GO TO " Ej.1.7.2"
  2 PRINT "AREA DE CILINDROS"
 10 PRINT "Longitud del radio: ";
 20 INPUT R
 30 PRINT R
 40 PRINT "Longitud de altura: ";
 50 INPUT A
 60 PRINT A
 70 PRINT "AREA LATERAL DEL CILINDRO: ";2*PI*R*A
```

```
80 PRINT
90 GO TO 10
```

```
 1 REM Ficha:" GO TO " Ej.1.7.2"
 2 PRINT "AREA DE CILINDROS"
10 PRINT "Longitud del radio: ";
20 INPUT R
30 PRINT R
40 PRINT "Longitud de altura: ";
50 INPUT A
60 PRINT A
70 PRINT "AREA TOTAL DEL CILINDRO: ";2*PI*R*A+4*PI*R
80 PRINT
90 GO TO 10
```

```
 1 REM Ficha:" GO TO " Ej.1.7.2"
 2 PRINT "SUPERFICIE DE LA ESFERA"
10 PRINT "Longitud del diametro: ";
20 INPUT D
30 PRINT D
40 PRINT "SUPERFICIE: ";PI*D^2
50 PRINT
60 GO TO 10
```

1.8.1.

```
 1 REM Ficha:" GO TO " Ej.1.8.1"
 2 PRINT "VOLUMEN DE PRISMAS"
10 PRINT "Area de la base: ";
20 INPUT S
30 PRINT S
40 PRINT "Altura de la figura: ";
50 INPUT A
60 PRINT A
70 PRINT "VOLUMEN: ";S*A
80 PRINT
90 GO TO 10
```

```
 1 REM Ficha:" GO TO " Ej.1.8.1"
 2 PRINT "VOLUMEN DE LA ESFERA"
10 PRINT "Longitud del diametro: ";
20 INPUT D
30 PRINT D
40 PRINT "VOLUMEN: ";PI/6*D^3
50 PRINT
60 GO TO 10
```

1.9.1.

```
 1 REM Ficha:" GO TO " Ej.1.9.1"
10 PRINT "VELOCIDAD DE UN MOVIL"
20 PRINT "Espacio recorrido : ";
30 INPUT E
40 PRINT E
50 PRINT "Tiempo empleado: ";
60 INPUT T
70 PRINT T
```

2.1.

```
  1 REM Ficha:" GO TO " Ej.2.1"
 10 PRINT "EDAD APROXIMADA DE UNA PERSONA"
 20 PRINT "Ano de nacimiento: ";
 30 INPUT A
 40 PRINT A
 50 PRINT "Ano actual: ";
 60 INPUT A1
 70 PRINT A1
 80 PRINT "Edad :";A1-A;" anos"
 90 PRINT
100 GO TO 20
```

IF/THEN	IF expresión THEN sentencia	SI ... / ENTONCES ...

INTERPRETACION:

SI el resultado de la expresión es *cierto* (o *falso*) **ENTONCES** obedece la orden dada por sentencia.

POSIBILIDADES:

La expresión siempre estará formada por operadores de relación u operadores de relación y operadores lógicos.

La sentencia puede ser cualquiera de las que figuran en el ESQUEMA DE COMANDOS BASIC.

FORMA DE TECLEAR LA INSTRUCCION

Teclear **IF, la expresión, THEN y la sentencia,** seguido de **ENTER.**

* Recordar:

- Que **la expresión** siempre debe contener operadores de relación u operadores lógicos y de relación.
- Que **la sentencia** puede ser cualquiera del BASIC (PRINT, GO TO, etc.).

EJEMPLO:

1. Para ser candidata a Miss Universo se debe medir más de 1,70 metros y pesar menos de 70 kilogramos.

   ¿Podría hacer un programa en el que dado el nombre, la estatura y el peso de cada aspirante se obtuviera la respuesta de si es o no admitida?

SOLUCION:

PROGRAMA	COMENTARIOS
1Ø INPUT "¿Nombre? "; N$	El programa se interrumpirá durante su ejecución hasta que se introduzca la cadena de caracteres que represente el nombre.
2Ø INPUT "¿Talla? "; T	Se producirá una nueva interrupción hasta que se introduzca el valor numérico correspondiente a la talla.
3Ø INPUT "¿Peso? "; P	Se producirá otra interrupción hasta que se teclee el valor numérico del peso.
4Ø IF T < 1.7 THEN GO TO 11Ø	SI el valor dado a T es menor que 1.7 ENTONCES, y sólo entonces, el programa salta a la línea 11Ø.

PROGRAMA	COMENTARIOS
5Ø IF P > 6Ø THEN GO TO 11Ø	SI el valor dado a la variable P es mayor que 6Ø ENTONCES, y sólo entonces, el programa salta a la línea 11Ø.
6Ø PRINT N$, "admitida"	Si el ordenador llega a leer esta instrucción es, evidentemente, porque en la línea 4Ø el valor de T era mayor que 1.7, y en la línea 5Ø el valor de P era menor que 60, con lo cual la candidata supera las condiciones impuestas de talla y peso. Por tanto, se ordena la impresión de su nombre y el mensaje *admitida.*
7Ø INPUT "¿Otra candidata? "; O$	Con esta interrupción, el programa queda a la espera de una cadena de caracteres, cuya función se ve en la línea siguiente.
8Ø IF O$ = "NO" THEN STOP	SI la cadena asignada a la variable O$ es igual a NO, entonces el programa se detiene, ya que así lo ordena la sentencia STOP.
9Ø CLS	Esta sentencia –que se explicará en otro momento– ordena un borrado de pantalla, con el que desaparecen todas las impresiones previas.
1ØØ GO TO 1Ø	Para que el ordenador llegue a leer esta instrucción habrá sido necesario que el valor dado a O$ haya sido distinto a NO y, de esta forma, habremos mandado la lectura del programa a la línea 1Ø, comenzando de nuevo el proceso.
11Ø PRINT N$, "rechazada"	Esta línea de programa sólo será leída si alguna de las líneas 4Ø ó 5Ø lo permiten.
12Ø GO TO 7Ø	En esta línea se provoca un salto incondicional a la línea 7Ø para saber si el proceso debe repetirse.

DIDACTICA:

* Recuerde a los alumnos:

– Que existen símbolos específicos para establecer una relación entre los términos de una expresión:

a) *Los operadores de relación,* que sólo operan en proposiciones entre dos operandos: =, < >, <, >, <=, >=.

b) *Los operadores lógicos,* que actúan entre operandos en los que, a su vez, intervienen operadores de relación. Son operadores lógicos AND, OR y NOT.

* Explique a los alumnos:

– Que el comando IF usado con THEN establece una determinada condición dentro del programa.

– Que cuando la condición establecida por IF se cumple, se produce el salto incondicional del programa impuesto por THEN.

– Que cuando la condición no se cumple, el programa continúa ejecutando la línea siguiente a la del comando IF.

– Que la expresión debe estar formada por operadores de relación u operadores de relación y operadores lógicos.

– Que el comando IF puede combinarse también con otros comandos BASIC

(PRINT, LET, STOP, etc.), por lo que la sentencia puede ser cualquiera de las que figura en el ESQUEMA DE COMANDOS BASIC (1).

* Que los alumnos observen experimentalmente:

– Cómo es necesario contestar a las cuestiones planteadas con textos exactos a los que están programados. En caso contrario –aunque se trate de ligeros errores o modificaciones (supresión o aumento de una letra; cambio de una mayúscula por una minúscula, o viceversa; etc.)– el computador "entenderá" que la respuesta es errónea.

– Cómo se combina el comando IF con otros comandos BASIC.

(1) Hay computadores que no exigen explícitamente la sentencia GO TO después de THEN. Así la instrucción IF x = 2 THEN 236∅, tendría para ese tipo de computadores la misma interpretación que IF x = 2 THEN GO TO 236∅.

EJERCICIOS:

1. Escribir un programa que diga si es o no correcta la suma (resta, multiplicación o división) de dos números cualesquiera.

2. Escribir un programa para contar (o descontar) de **n** en **n** desde un número **A** (*).

3. Escribir un programa para realizar operaciones combinadas (*).

Por ejemplo:

A + B – C
(A + B) * C
(A + B – C)/D

4. Escribir un programa para resolver problemas de este tipo:

Un individuo tiene **P** pesetas y quiere comprar **U** unidades de una mercancía, a **X** pesetas la unidad, y **V** unidades de otra mercancía, a **Y** pesetas la unidad. ¿Cuánto le costará cada mercancía, cuánto importará la compra y cuánto le sobrará?

5. Escribir programas que comparen entre sí:

– el valor de varios números.
– el valor de varias letras.
– el valor de varias palabras.

6. Escribir el programa que, al introducir las variables **b** (base) y **h** (altura), nos diga si es o no correcto el cálculo que hagamos del área **S** de un paralelogramo (*).

7. Escribir el programa que, al introducir las variables **b** (base) y **h** (altura), nos diga si es o no correcto el cálculo que hagamos del área **S** de un triángulo (*).

8. Escribir el programa que, al introducir las variables **lb** (lado de la base), **ab** (apotema de la base) y **H** (altura del prisma), nos diga si son o no correctos los cálculos que hagamos del área lateral **S1**, el área total **St** y el volumen **V** de un prisma (*).

9. Escribir el programa que, al introducir la variable **r** (radio), nos diga si son o no correctos los cálculos que hagamos del área **S** y el volumen **V** de una esfera (*).

10. Escribir el programa que nos diga si es correcta o no la relación que establezcamos entre cada país y su capital.

Países: España, Portugal, Francia, Italia, Albania, Grecia.
Capitales: Madrid, Lisboa, París, Roma, Tirana, Atenas.

11. Escribir el programa que nos diga si es correcta o no la relación que establezcamos entre cada órgano del aparato digestivo humano con el acto que realiza (*).

Organos: boca, dientes, glándulas salivales, faringe, esófago, estómago, intestino delgado, intestino grueso.

Actos: ingestión, masticación, insalivación, deglución, progresión, quimificación, quilificación, absorción, defecación.

12. Escribir el programa para calcular la nota final **NF** de un alumno **X**, cuyas evaluaciones a lo largo del curso han sido **N1**, **N2** y **N3**.

Calificaciones posibles:

I: insuficiente < 5
SF: suficiente = 5
B: bien = 6
NT: notable = 7 y 8
SB: sobresaliente = 9 y 10

13. Idear sus propios programas utilizando los comandos conocidos: PRINT, LET, INPUT, GO TO e IF/THEN o IF/THEN GO TO.

SOLUCIONES

1.

```
  1 REM Ficha:" IF  THEN " Ej.1"
 10 PRINT "COMPROBACION SUMA"'
 20 PRINT "TOTAL: ";
 30 INPUT T
 40 PRINT T
 50 PRINT "Sumando: ";
 60 INPUT N
 70 PRINT N
 80 PRINT "Sumando: ";
 90 INPUT N1
100 PRINT N1,
110 IF N+N1<>T THEN  PRINT "Incorrecta"
120 IF N+N1=T THEN  PRINT "Correcta"
130 PRINT
140 GO TO 20
```

4.

```
  1 REM Ficha:" IF / THEN  " Ej.4"
 10 PRINT "Dinero que tiene: ";
 20 INPUT P: PRINT P
 30 PRINT "Unidades ""A""";
 40 INPUT U: PRINT U'"Precio: ";
 50 INPUT X: PRINT X'"Unidades ""B""";
 60 INPUT V: PRINT V'"Precio: ";
 70 INPUT Y: PRINT Y
 80 PRINT "COSTO MERCANCIA ""A"": ";U*X
 90 PRINT "COSTO MERCANCIA ""B"": ";V*Y
100 PRINT "COSTO TOTAL: ";U*X+V*Y
110 LET D=U*-X+V*-Y+P: PRINT
120 IF D>0 THEN  PRINT "SOBRAN: ";D
130 IF D<0 THEN  PRINT "FALTAN: ";D
140 PRINT
150 GO TO 10
```

5.

```
  1 REM Ficha:" IF / THEN " Ej.5"
 10 PRINT "COMPARA Y ORDENA NUMEROS"
 20 PRINT "Escriba: ";
 30 INPUT A;" y ";B;" y ";C
 40 PRINT A;" ";B;" ";C
 50 IF A>B THEN  LET N=A: LET A=B: LET B=N
 60 IF A>C THEN  LET N=A: LET A=C: LET C=N
 70 IF B>C THEN  LET N=B: LET B=C: LET C=N
 80 PRINT A'B'C
 90 PRINT
100 GO TO 20
```

```
  1 REM Ficha:" IF / THEN " EJ.5"
 10 PRINT "COMPARA LETRAS Y PALABRAS"
 20 PRINT "Escriba: ";
 30 INPUT A$;B$;C$
 40 PRINT A$;" ";B$;" ";C$
 50 IF A$>B$ THEN  LET N$=A$: LET A$=B$: LET B$=N$
 60 IF A$>C$ THEN  LET N$=A$: LET A$=C$: LET C$=N$
 70 IF B$>C$ THEN  LET N$=B$: LET B$=C$: LET C$=N$
 80 PRINT A$;" ";B$;" ";C$
 90 PRINT
100 GO TO 20
```

10.

```
  1 REM Ficha:" IF / THEN " Ej.10"
 10 PRINT "RELACION ENTRE NACIONES"'"Y SUS CAPITALES."''
 20 LET A$="ESPANA": PRINT "1 "+A$,
 30 LET H$="PARIS": PRINT "a) "+H$
 40 LET B$="PORTUGAL": PRINT "2 "+B$,
 50 LET I$="ATENAS": PRINT "b) "+I$
 60 LET C$="FRANCIA": PRINT "3 "+C$,
 70 LET J$="TIRANA": PRINT "c) "+J$
 80 LET D$="ITALIA": PRINT "4 "+D$,
 90 LET K$="MADRID": PRINT "d) "+K$
100 LET E$="ALBANIA": PRINT "5 "+E$,
110 LET L$="LISBOA": PRINT "e) "+L$
120 LET F$="GRECIA": PRINT "6 "+F$,
130 LET M$="ROMA": PRINT "f) "+M$
140 PRINT '"Nacion:","Capital:"''
150 INPUT N$
160 IF N$="1" THEN  PRINT A$,K$
170 IF N$="2" THEN  PRINT B$,L$
180 IF N$="3" THEN  PRINT C$,H$
190 IF N$="4" THEN  PRINT D$,M$
200 IF N$="5" THEN  PRINT E$,J$
210 IF N$="6" THEN  PRINT F$,I$
220 IF N$="a" THEN  PRINT C$,H$
230 IF N$="b" THEN  PRINT F$,I$
240 IF n$="c" THEN  PRINT E$,J$
250 IF N$="d" THEN  PRINT A$,K$
260 IF N$="e" THEN  PRINT B$,L$
```

```
270 IF N$="f" THEN  PRINT F$,I$
280 GO TO 150
```

12.
```
  1 REM Ficha:" IF / THEN " Ej.12"
 10 PRINT '"1a.Evaluacion: ";
 20 INPUT N1
 30 PRINT N1
 40 PRINT "2a.Evaluacion: ";
 50 INPUT N2
 60 PRINT N2
 70 PRINT "3a.Evaluacion: ";
 80 INPUT N3
 90 PRINT N3''
100 LET NF=(N1+N2+N3)/3: IF NF>10 THEN  GO TO 10
110 PRINT "NOTA FINAL: ";NF,
120 IF NF<5 THEN  PRINT "Insuficiente": GO TO 10
130 IF NF<6 THEN  PRINT "Suficiente": GO TO 10
140 IF NF<7 THEN  PRINT "Bien": GO TO 10
150 IF NF<9 THEN  PRINT "Notable": GO TO 10
160 IF NF<=10 THEN  PRINT "Sobresaliente": GO TO 10
```

FOR... TO/NEXT FOR... TO... STEP/NEXT		PARA ... HASTA/PROXIMO PARA ... HASTA ... SALTO/PROXIMO

FOR variable = {expresión numérica 1} TO {expresión numérica 2} /NEXT variable

INTERPRETACION:

La variable va cambiando de valor –de unidad en unidad– partiendo del valor dado por la expresión numérica 1, y hasta alcanzar el valor dado por la expresión numérica 2.

El cambio de valor de la variable se producirá –normalmente, de unidad en unidad– cuando en la secuencia normal de lectura se encuentre la sentencia

NEXT variable

Con ella, el programa cambia su secuencia de lectura a la línea donde se encuentra la sentencia inicial FOR ... TO.

POSIBILIDADES:

El cambio de valor de la variable puede ser distinto de la unidad. Para ello, bastará con añadir la sentencia **STEP seguida de la expresión numérica 3** a la sentencia FOR ... TO, donde la expresión numérica 3 determinará el salto que ha de dar la variable al llegar a NEXT, quedando la estructura de la sentencia así:

FOR variable = {expres. numér.1} TO {expres. numér.2} STEP {expres. numér.3} /NEXT variable

En esta secuencia, la variable que sigue a FOR sólo puede ser una letra.

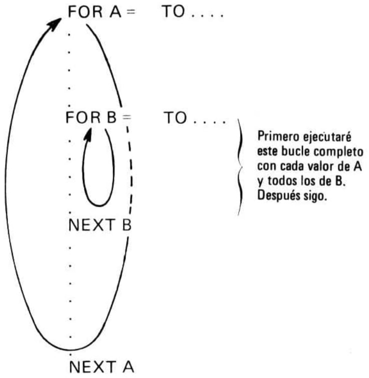
Esta sentencia produce un *bucle de lectura* que se repite, una y otra vez, hasta que la variable alcanza el valor de la expresión numérica 2, pudiéndose representar esquemáticamente de la siguiente forma:

Bucle FOR ... TO/NEXT normal (de unidad en unidad)

n° de línea	**instrucciones**
n	FOR variable = {expres. numér.1} TO {expres. numér.2}
n + 1	
n + 2	
. . .	
n + n	NEXT variable

Bucle FOR ... TO ... STEP/NEXT unidades (de m en m unidades)

n° de línea	**instrucciones**
n	FOR variable 1 = {expres. numér.1} TO {expres. numér.2}
n + 1	
n + 2	
n + 3	FOR variable 2 = {expres. numér.3} TO {expres. numér.4}
. . .	
. . .	Primero se ejecuta este *bucle* completo, por cada
n + m	valor de la variable 1 y todos los de la variable 2
n+m+1	
n+m+2	NEXT variable 2
n+m+3	
. . . . .	
. . . . .	
n+m+1	NEXT variable 1



La secuencia normal de lectura se restablece cuando la variable 1 se hace igual a la expresión numérica 2. En ese momento, la siguiente línea a leer es aquélla que sigue a la sentencia NEXT.

FORMA DE TECLEAR LAS INSTRUCCIONES

Teclear **FOR, el nombre de la variable numérica, el signo igual (=), la expresión numérica 1, TO y la expresión numérica 2**, seguido de **ENTER**, para que el valor de la variable cambie *de unidad en unidad.*

Teclear **FOR, el nombre de la variable numérica, el signo igual (=) la expresión numérica 1, TO, la expresión numérica 2, STEP y la expresión numérica 3**, seguido de **ENTER,** para que el valor de la variable cambie *de n en n unidades* indicadas en la expresión numérica 3.

Secuencia de instrucciones del bucle.

Teclear **NEXT y el nombre de la variable numérica,** seguido de ENTER.

EJEMPLO:

1. Confeccionar un programa que, una vez ejecutado, imprima en pantalla los resultados de la tabla de multiplicar por 2 desde Ø hasta 1Ø.

SOLUCION:

PROGRAMA	COMENTARIOS
1Ø FOR Y = Ø TO 1Ø	En este programa nombramos la variable con la letra Y; la expresión numérica 1 la hacemos igual a Ø, y la expresión numérica 2 la hacemos igual a 1Ø.
2Ø LET A = 2 * Y	En la línea 2Ø, damos a la variable A el valor que resulte de multiplicar por 2 el valor actual de la

	variable Y –que tomará sucesivamente los valores Ø, 1, 2, 3, 4, 5, 6, 7, 8, 9 y 1Ø–.
3Ø PRINT A	**La línea 3Ø** obliga a imprimirse el valor actual de A.
4Ø NEXT Y	**Al llegar a** la línea 4Ø, el programa seguirá su lectura en la línea 1Ø, siempre y cuando el valor de Y no haya llegado a 1Ø. En otro caso, pasaría a la línea 5Ø y el programa parará.
5Ø STOP	

EJEMPLO:

2. Escribir un programa que, una vez ejecutado, imprima en pantalla los números impares del 1 al 5Ø.

SOLUCION:

PROGRAMA	COMENTARIOS
1Ø FOR X = 1 TO 5Ø STEP 2 **2Ø PRINT X** **3Ø NEXT X**	En este caso, obligamos a saltar a la variable X de dos en dos –gracias a STEP 2–, partiendo del valor 1 y hasta llegar al valor 5Ø.

DIDACTICA:

* Explique a los alumnos:
  - Que hay comandos BASIC que no son imprescindibles para programar, pero que su utilización ahorra tiempo y trabajo, haciendo más fácil la programación. Entre estos comandos están FOR ... TO/NEXT y FOR ... TO ... STEP/ NEXT.
  - Que el comando FOR es el primero de un *bucle*, que se cierra con el comando NEXT.
  - Que *los bucles* se utilizan para repetir, a lo largo de un programa, una o varias operaciones iguales.
  - Que un *bucle* FOR ... TO se inicia cuando la variable situada a continuación de FOR toma el valor inicial de la expresión numérica 1; va cambiando de unidad en unidad si no hay STEP, y termina cuando alcanza (TO) el valor final de la expresión numérica 2.
  - Que el cambio de valor de la variable se produce cada vez que, en la secuencia normal de lectura, se encuentra la sentencia final NEXT, con lo que el programa cambia su secuencia lectora a la línea donde se encuentra la sentencia inicial FOR...TO, para iniciar otra vez la secuencia de lectura normal.
  - Que el cambio de valor de la variable puede ser distinto de la unidad, según la voluntad del programador. Para conseguirlo, basta con añadir a la sentencia inicial FOR ... TO la sentencia STEP, en la que esta expresión determinará el incremento (o la disminución) de valor de la variable definida con FOR al llegar, en la secuencia normal de lectura, a la sentencia final NEXT.

* Las explicaciones anteriores son bastante complejas, por lo que el profesor debe adaptarlas a la mentalidad de sus alumnos, e incluso, suprimirlas en muchos casos.

* Que los alumnos observen experimentalmente:

- Cómo el valor inicial de la variable se incrementa (o disminuye) en un número entero o fraccionario, positivo o negativo, cuando se utiliza FOR...TO... STEP/NEXT.
- Como el computador continúa el programa en la línea siguiente a la sentencia NEXT después de finalizar un *bucle*.
- Qué ocurre cuando dentro de una sentencia FOR...TO/NEXT o FOR...TO... STEP/NEXT se incluye una instrucción que obliga al programa a salirse del *bucle (1)*.
- Cómo se puede imprimir en pantalla una columna con los valores que va tomando la variable.

(1) No es buena costumbre dejar bucles FOR ... TO sin agotar o, en todo caso, no tener el valor de la variable bajo control.

EJERCICIOS

1. Escribir un programa que, ejecutado, imprima en pantalla los números del Ø al 1ØØ.

2. Escribir el programa que, ejecutado, imprima en pantalla diez veces la palabra *"pollo"* en una columna, y otro programa que, al ejecutarlo, imprima en pantalla diez veces seguidas la palabra *"pío"*.

3. Escriba el programa que, ejecutado, imprima en pantalla en diez líneas y dos columnas diez veces la palabra *"pollo"* y diez veces la palabra *"pío"*.

4. Escribir los programas que, una vez ejecutados, impriman en pantalla:
   - 4.1. Diez veces seguidas la palabra *"pollo"* y otras diez veces seguidas la palabra *"pío"*. (*).
   - 4.2. Diez veces en eolumna la palabra *"pollo"* y otras diez veces en la misma columna la palabra *"pío"* (*).
   - 4.3. Cinco veces la palabra *"pollo"* y cinco veces la palabra *"pío"* en dos columnas (*).
   - 4.4. Diez veces la palabra *"pollo"* en una columna y diez veces la palabra *"pío"* en otra columna, de modo que haya veinte líneas en pantalla en total (*).

5. Escriba el programa que, ejecutado, imprima en pantalla la tabla de sumar 5 desde Ø hasta 1Ø.

6. Escriba el programa que, ejecutado, imprima en pantalla las tablas de sumar 1, 2, 3, ..., 1Ø desde Ø hasta 1Ø

7. Escriba el programa que, ejecutado, imprima en pantalla los números pares menores que 1ØØ.

8. Escriba el programa que, ejecutado, imprima en pantalla los múltiplos de 5 desde Ø hasta 1ØØ

9. Escribir el programa que, dándole un número N del Ø al 1Ø, nos diga si es correcto o no el producto de N por otro número M también del Ø al 1Ø.

10. Idear sus propios programas utilizando los comandos PRINT, LET, INPUT, FOR ... TO/NEXT, FOR ... TO ... STEP/NEXT, IF y GO TO.

SOLUCIONES

1.

```
 1 REM Ficha:" FOR / TO / NEXT " Ej.1
10 PRINT "NUMEROS"
20 FOR N=1 TO 20
```

```
30 PRINT N;"     ";n+20;"     ";N+40;"     ";N+60;"     ";
   N+80
40 NEXT N
```

2.

```
 1 REM Ficha:" FOR / TO : NEXT " Ej.2"
10 FOR N=1 TO 10: PRINT "Pollo": NEXT N
20 FOR N=1 TO 10: PRINT "Pio": NEXT N
```

3.

```
 1 REM Ficha:" FOR / TO : NEXT " Ej.3"
10 FOR N=1 TO 10: PRINT "Pollo","Pio": NEXT N
```

5.

```
 1 REM Ficha:" FOR / TO : NEXT " Ej.5"
10 PRINT "TABLA DE SUMAR 5"
20 FOR N=0 TO 10
30 PRINT N;" mas ";5;" = ";N+5
40 NEXT N
```

6.

```
 1 REM Ficha:" FOR / TO : NEXT " Ej.6"
10 PRINT "TABLA DE SUMAR": LET A=1
20 FOR N=0 TO 10
30 PRINT N;" + ";A;" = ";N+A
40 NEXT N
50 LET A=A+1: IF A>10 THEN  STOP
60 GO TO 20
```

7.

```
 1 REM Ficha:" FOR / TO : NEXT " Ej.7"
10 PRINT "NUMEROS PARES"
20 FOR N=2 TO 98 STEP 2
30 PRINT N;"  ";
40 NEXT N
```

8.

```
 1 REM Ficha:" FOR / TO : NEXT " Ej.8"
10 PRINT "MULTIPLOS DE 5"
20 FOR N=1 TO 20
30 PRINT N*5;" ";
40 NEXT N
```

9.

```
  1 REM Ficha:" FOR / TO : NEXT " Ej.9"
 10 PRINT "Di un numero del 1 al 10"
 20 INPUT N: PRINT N
 30 FOR I=1 TO 10
 40 PRINT N;"*";I;"=";
 50 INPUT M: PRINT M
 60 IF M=N*I THEN  GO TO 90
 70 PRINT "No, intentalo otra vez"
 80 GO TO 40
 90 PRINT "SI"
100 NEXT I
110 STOP
```

GO SUB/RETURN	GO SUB número de línea	IR A SUBRUTINA/VOLVER

INTERPRETACION:

La lectura del programa salta incondicionalmente a un número de línea y continúa a partir de ella hasta encontrar la sentencia

RETURN

con la que se produce un salto incondicional a la línea de programa siguiente a aquélla que contenía la sentencia GO SUB anterior.

INTERPRETACION:

Estas sentencias son utilizadas cuando una parte de un programa se repite varias veces y, consiguientemente, es más cómodo considerar esa parte del programa como una *subrutina* a la que poder dirigirse –gracias a GO SUB– tantas veces como sea necesario.

También se suele utilizar GO SUB para organizar un programa en *subrutinas*, de tal forma, que su estructura sea clara.

FORMAS DE TECLEAR LAS INSTRUCCIONES

Teclear **GO SUB y el número de línea**, seguido de **ENTER**, para que la secuencia de lectura del programa se repita desde *la línea del número dado* y hasta encontrar **RETURN**.

Teclear **RETURN,** seguido de **ENTER,** para seguir la lectura del programa desde la *línea siguiente* de la que contiene el **GO SUB** anterior.

EJEMPLO:

1. Consideremos nuevamente el programa para elegir candidatas a Miss Universo y enfoquémoslo desde el punto de vista de las subrutinas.

SOLUCION:

PROGRAMA	COMENTARIOS
2 INPUT "¿Alguna candidata? "; O$	En el programa anterior, introducimos las líneas 2, 4. 6 y 8.
4 IF O$ = "SI" THEN GO SUB 1Ø	Si el valor de O$ es SI, ENTONCES el programa se dirige a la subrutina que comienza en la línea 1Ø. Pero si el valor de O$ es NO, ENTONCES el programa termina
6 IF O$ = "NO" THEN STOP	

PROGRAMA	COMENTARIOS

```
8 GO TO 2
1Ø  INPUT "¿Nombre? "; N$
2Ø INPUT "¿Talla? "; T
3Ø INPUT "¿Peso?  "; P
4Ø IF T 1.7 THEN GO TO 11Ø
5Ø IF P 6Ø THEN GO TO 11Ø
6Ø PRINT N$, "admitida"
7Ø RETURN
9Ø CLS
1ØØ GO TO 1Ø
11Ø PRINT N$, "rechazada"
12Ø RETURN
```

En el programa anterior, borramos las líneas 7Ø, 8Ø y 12Ø. Para hacerlo, basta con teclear cada uno de estos tres números seguido de ENTER, con lo que cada línea desaparecerá automáticamente del listado.
Añadimos las líneas:
7Ø RETURN y
12Ø RETURN
También podíamos haber tecleado estas dos líneas directamente, con lo que hubieran sustituido a las que teníamos antes sin necesidad de borrarlas.
Dejamos el resto del programa como estaba.

DIDACTICA:

* Recuerde a los alumnos:

- Que las expresiones numéricas están constituidas sólo por números, mientras que las expresiones alfanuméricas pueden estar constituidas por grupos de palabras, de números o de palabras y números.
- Cómo se borra una línea de programa y cómo se sustituye por otra.
- El empleo del comando GO TO.

* Explique a los alumnos:

- Que, muchas veces, dentro de un *programa general,* hay una o más líneas que se repiten una y otra vez, y que estas líneas podemos considerarlas como un *programa parcial* que se repite.
- Que, para escribir una sola vez este *programa parcial* y poder utilizarlo dentro del *programa general* tantas veces como sea necesario, existe en BASIC el comando GO SUB.
- Que GO SUB es la abreviatura en inglés de "go subrutine", que significa "ve a la subrutina".
- Que un *programa general* es una **rutina**, ya que, al ejecutarlo, repite siempre el mismo proceso.
- Que un *programa parcial* es una **subrutina** que está dentro de un *programa general* o *rutina.*
- Que, cuando en la secuencia normal de lectura de un programa, el ordenador encuentra GO SUB, salta incondicionalmente a la línea indicada y continúa la lectura a partir de ella hasta que encuentra la sentencia RETURN, que significa "volver".
- Que al encontrar la sentencia RETURN, el ordenador vuelve a la lectura de la línea siguiente a aquélla donde estaba GO SUB.
- Que este proceso puede repetirse tantas veces como sea necesario utilizar la *subrutina.*

* Que los alumnos observen experimentalmente:

- Cómo el empleo de GO SUB/RETURN no es imprescindible, pero resulta muy útil para abreviar los programas.

– Las diferencias entre el empleo de GO TO y GO SUB/RETURN.
– La conveniencia de utilizar la sentencia STOP para detener la ejecución del programa al llegar a la línea donde el programa general termina.
– Cómo poner de nuevo en marcha un programa, detenido por un STOP, utilizando el comando CONTINUE.

EJERCICIOS:

Escribir los siguientes programas utilizando *subrutinas:*

1. El que nos diga si es o no correcta la suma (resta, multiplicación o división) de dos números cualesquiera.

2. El que sirva para realizar determinadas operaciones combinadas (*). Por ejemplo:

   A * B – C + D
   A * (B – C) + D
   (A * B) – (C + D)
   (A * B – C)/D

3. El que sirva para resolver problemas de este tipo:

   Dos individuos A y B van de compras. A tiene P pesetas y compra U unidades de una mercancía, a X pesetas la unidad, y V unidades de otra mercancía, a Y pesetas la unidad. B tiene P1 pesetas y compra U1 unidades de una mercancía, a X1 pesetas la unidad, y V1 unidades de otra mercancía, a Y1 pesetas la unidad. ¿Cuánto importará la compra de cada individuo y cuánto dinero le sobrará?

4. El que, al introducir las variables de la base y la altura, nos diga si es correcto o no el cálculo que hagamos del área de un paralelogramo (*).

5. El que, al introducir las variables de la base y la altura, nos diga si es correcto o no el cálculo que hagamos del área de un triángulo (*).

6. El que, al introducir las variables del lado de la base, la apotema de la base y la altura de un prisma, nos diga si es correcto o no el cálculo que hagamos del área lateral, el área total y el volumen (*).

7. El que, al introducir la variable del radio, nos diga si es correcto o no el cálculo que hagamos del área y el volumen de una esfera (*).

8. El que nos diga si es correcta o no la relación que establezcamos entre cada país y su capital, y separe cada respuesta con una línea de asteriscos.
   (Utilizar los países y capitales propuestos en el ejercicio 10 del capítulo 5).

9. El que nos diga si es correcta o no la relación que establezcamos entre cada órgano del aparato digestivo humano y el acto que realiza, y separe las respuestas por una línea de .–.–.–.– (*).

10. El que calcule la nota final de los alumnos de una clase, sabiendo las notas de las tres evaluaciones del curso escolar (*).

11. Idear sus propios programas utilizando subrutinas,

SOLUCIONES

1.

```
  1 REM Ficha:" GO SUB : RETURN " Ej.1"
 10 GO SUB 700: PRINT "Total hallado: ";: INPUT T: PRINT T
```

```
 20 PRINT "Operando primero: ";: INPUT A: PRINT A
 30 PRINT "Operando segundo: ";: INPUT B: PRINT B
 40 PRINT "Signo de la operacion: ";: INPUT O$
 50 IF O$="+" THEN  PRINT "Suma": GO TO 100
 60 IF O$="-" THEN  PRINT "Resta": GO TO 200
 70 IF O$="*" THEN  PRINT "Multiplicacion": GO TO 300
 80 IF O$="/" THEN  PRINT "Division": GO TO 400
 90 PRINT "OPERACION NO PROGRAMADA"'': GO TO 10
100 IF T=A+B THEN  GO TO 500
110 GO TO 600
200 IF T=A-B THEN  GO TO 500
210 GO TO 600
300 IF T=A*B THEN  GO TO 500
310 GO TO 600
400 IF T=A/B THEN  GO TO 500
410 GO TO 600
500 PRINT "CORRECTO"'': GO TO 10
600 PRINT "INCORRECTO": GO TO 10
700 FOR N=0 TO 31: PRINT "*";: NEXT N: RETURN
```

3.

```
  1 REM Ficha:" GO SUB : RETURN " Ej.3"
 10 PRINT "Dinero de ""A"": ";: INPUT P: PRINT P: GO
    SUB 100
 20 LET U=Ud: LET X=Pr: GO SUB 100
 30 LET V=Ud: LET Y=Pr
 40 PRINT "TOTAL COMPRA DE""A"": ";U*X+V*Y'"DIFERENCIA:
    ";P-U*X-V*Y
 60 PRINT "Dinero de ""B"": ";: INPUT P1: PRINT P1: GO
    SUB 100
 70 LET U1=Ud: LET X1=Pr: PRINT "Valor: ";U1*X1: GO SUB
    100
 80 LET V1=Ud: LET Y1=Pr
 90 PRINT "TOTAL COMPRA DE ""B"": ";U1*X1+V1*Y1'"DIFERE
    NCIA: ";P1-U1*X1-V1*Y1''
 95 STOP
100 PRINT "Unidades: ";: INPUT ud: PRINT Ud'"Precio: ";
110 INPUT Pr: PRINT Pr'"Valor: ";Ud*Pr
120 RETURN
```

8.

```
  1 REM Ficha:" GO SUB : RETURN " Ej.8"
 10 PRINT "RELACION ENTRE NACIONES"'"Y SUS CAPITALES."''
 20 LET A$="ESPANA": PRINT "1 "+A$,
 30 LET H$="PARIS": PRINT "a) "+H$
 40 LET B$="PORTUGAL": PRINT "2 "+B$,
 50 LET I$="ATENAS": PRINT "b) "+I$
 60 LET C$="FRANCIA": PRINT "3 "+C$,
 70 LET J$="TIRANA": PRINT "c) "+J$
 80 LET D$="ITALIA": PRINT "4 "+D$,
 90 LET K$="MADRID": PRINT "d) "+K$
100 LET E$="ALBANIA": PRINT "5 "+E$,
110 LET L$="LISBOA": PRINT "e) "+L$
120 LET F$="GRECIA": PRINT "6 "+F$,
```

```
130 LET M$="ROMA": PRINT "f)  "+M$
140 PRINT '"Nacion:","Capital:"''
150 INPUT N$: IF N$<>"" THEN  GO SUB 300
160 IF N$="1" THEN  PRINT A$,K$
170 IF N$="2" THEN  PRINT B$,L$
180 IF N$="3" THEN  PRINT C$,H$
190 IF N$="4" THEN  PRINT D$,M$
200 IF N$="5" THEN  PRINT E$,J$
210 IF N$="6" THEN  PRINT F$,I$
220 IF N$="a" THEN  PRINT C$,H$
230 IF N$="b" THEN  PRINT F$,I$
240 IF n$="c" THEN  PRINT E$,J$
250 IF N$="d" THEN  PRINT A$,K$
260 IF N$="e" THEN  PRINT B$,L$
270 IF N$="f" THEN  PRINT F$,I$
280 GO TO 150
300 FOR N=0 TO 31: PRINT "*";: NEXT N: RETURN
```

READ/DATA		LEER/DATOS

DATA	DATA constantes	DATOS

INTERPRETACION:

Con esta sentencia se almacenan valores –numéricos o alfanuméricos– a los cuales accede el programa secuencialmente, gracias a la sentencia **READ**, que estudiaremos después.

POSIBILIDADES:

Cada constante, dentro de la sentencia DATA, debe ir separada de la siguiente por una coma, y las constantes alfanuméricas deben escribirse entre comillas.

Las sentencias DATA son interpretadas por el BASIC como **almacén de datos** y, consiguientemente, no implican ningún tipo de operación al ser leídas. Esto quiere decir que las sentencias DATA se pueden colocar en cualquier parte del programa, sin que afecte al desarrollo del mismo; no obstante, es aconsejable colocar todos los DATA en un mismo sector del programa.

Un DATA puede contener tantas constantes como requiera el programador, y un programa puede tener tantos DATA como sean necesarios.

La sentencia DATA, para que sea operativa, necesita de la sentencia READ.

FORMA DE TECLEAR LA INSTRUCCION

Teclear **DATA y las constantes separadas por comas**, seguido de **ENTER**.

* Recordar que las constantes pueden ser valores numéricos y/o alfanuméricos. Por tanto, deben teclearse entre comillas (") cuando son alfanuméricos.

EJEMPLO:

Almacene en la línea de programa 1ØØØ los nombres de cinco capitales europeas.

SOLUCION:

1ØØØ DATA "Madrid", "París", "Londres", "Roma", "Dublín".

READ/DATA		LEER/DATOS

READ	READ variables	LEER

INTERPRETACION:

Con estas sentencias se leen, secuencialmente, las constantes almacenadas en las sentencias DATA, asignando estos valores –uno tras otro– a las variables. Estas variables van separadas por comas.

POSIBILIDADES:

Cuando la lectura de un programa llega a una sentencia READ, lee la primera constante aún no leída de todos los valores almacenados en todas las sentencias DATA del programa, asignando dicho valor a la variable correspondiente.

Las variables deben ser del mismo tipo –numéricas o alfanuméricas– que las constantes que lee de los DATA. En caso contrario, un mensaje de error avisará de la anomalía.

Una sentencia READ puede acceder a uno o varios DATA, o varios READ a un solo DATA, pero siempre secuencialmente.

Si la cantidad de variables situada tras una sentencia READ es mayor que el total de constantes, saldrá en la pantalla un mensaje de error advirtiendo que no hay suficientes constantes. Este mensaje suele ser OUT OF DATA, indicando a continuación el correspondiente número de línea.

Si hay más constantes en las sentencias DATA que variables en las sentencias READ, el excedente de constantes queda guardado hasta que otras sentencias READ obliguen a utilizarlo. Es decir, las variables de la siguiente sentencia READ tomarán los valores aún no leídos del total de las constantes de las sentencias DATA.

Para comenzar a leer las constantes desde el primer DATA hay que usar la sentencia *RESTORE*, como veremos posteriormente.

FORMA DE TECLEAR LA INSTRUCCION

Teclear **READ y el nombre de la variable o las variables separadas por comas y en el mismo orden que están en DATA,** seguido de **ENTER.**

EJEMPLO:

1. Escriba el programa, que una vez ejecutado, imprima en pantalla las cinco ciudades europeas del ejemplo anterior.

SOLUCION 1:

PROGRAMA	COMENTARIOS
1Ø READ A$, B$, C$, D$, E$	En la línea 1Ø damos valores a las variables indicadas, extrayéndolas secuencialmente del DATA de la línea 1ØØØ.
2Ø PRINT A$, B$, C$, D$, E$	En la línea 2Ø se ordena su impresión.
1ØØØ DATA "Madrid", "París", "Londres", "Roma", "Dublín"	

SOLUCION 2:

PROGRAMA	COMENTARIOS
1Ø FOR X = 1 TO 5 **2Ø READ V$** **3Ø PRINT V$** **4Ø NEXT X** **1ØØØ DATA "Madrid", "París"** **1Ø1Ø DATA "Londres", "Roma", "Dublín"**	En esta ocasión, una sola variable V$ va tomando sucesivamente los valores contenidos en los DATA.

EJEMPLO:

2. Escriba el programa que, una vez ejecutado, imprima en pantalla el nombre de cinco alumnos y sus calificaciones.

SOLUCION:

PROGRAMA	COMENTARIOS
1Ø FOR X = 1 TO 5 **2Ø READ V$: PRINT V$** **3Ø READ V: PRINT V.** **4Ø NEXT X** **1ØØ DATA "JUAN", 5, "José", 6, "Ricardo", 4, "Luis", 8, "Tomás", 9.**	Observe cómo la secuencia de lectura de los READ exige que los valores de los DATA sean, alternativamente, alfanuméricos y numéricos.

EJEMPLO:

3. Escriba el programa que, una vez ejecutado, imprima en pantalla la suma de los números pares y la suma de los números impares menores que 1Ø.

SOLUCION:

PROGRAMA	COMENTARIOS
1Ø DATA 2, 4, 6, 8, 1, 3, 5, 7, 9, "pares < 1Ø", "impares < 1Ø"	Las sentencias READ de las líneas 2Ø, 3Ø y 5Ø van tomando los valores que hay en la sentencia

PROGRAMA	COMENTARIOS
2∅ READ A, B, C, D 3∅ READ E, F, G, H, I 4∅ LET S = A + B + C + D 5∅ LET T = E + F + G + H + I 6∅ READ A$, B$ 7∅ PRINT "La suma de los números "; A$; " es "; S; " y la suma de los números "; B$; " es "; T	DATA de la línea 1∅. Observe que las variables A$ y B$ de la línea 6∅, que son alfanuméricas, toman los valores escritos entre comillas en la línea 1∅. Recuerde: No hay problema cuando hay más constantes que variables. Las constantes que sobran quedan a la espera de que otra u otras sentencias READ los ordene leer.

DIDACTICA:

* Recuerde a los alumnos:
  - Que hay que separar las constantes o las variables de las sentencias DATA o READ, respectivamente, mediante comas.
  - Que las expresiones alfanuméricas hay que teclearlas entre comillas.
  - Que las variables alfanuméricas se representan siempre seguidas del símbolo $.
* Explique a los alumnos:
  - Que podemos considerar READ como un "almacén" de **variables** numéricas y/o alfanuméricas escritas en una misma línea de programa, separadas por comas.
  - Que DATA es también como un "almacén" de **constantes** numéricas y/o alfanuméricas escritas en una misma línea, separadas por comas, así mismo.
  - Que las variables que hay en READ deben ser del mismo tipo –numéricas o alfanuméricas– que las correspondientes constantes que hay en DATA. Si no es así, en la pantalla aparece un mensaje de error que nos avisa de la anomalía.
  - Que las sentencias READ y DATA no son imprescindibles para programar, pero que son unos auxiliares que permiten ahorrar gran parte del tiempo que llevaría escribir varias sentencias LET y/o INPUT.
  - Que el ordenador, en la secuencia normal de lectura del programa, no tiene en cuenta los datos almacenados en DATA hasta que no encuentra la sentencia READ.
  - Que, entonces, la sentencia READ le obliga a leer, una tras otra y en el mismo orden en que están almacenadas, las constantes que hay en DATA, asignando estos valores, también en el mismo orden y uno tras otro, a las variables que hay en READ.
  - Que una sentencia READ puede acceder a una o varias sentencias DATA, secuencialmente.
  - Que varias sentencias READ pueden acceder a una o varias sentencias DATA, secuencialmente.
  - Que en un programa puede haber más constantes en DATA que variables en READ, pero nunca puede haber más variables en READ que constantes en DATA. En este último caso, el ordenador nos avisará del error.
* Que los alumnos observen experimentalmente:
  - Cómo se abrevia un programa que contiene varias sentencias LET y/o INPUT utilizando en su lugar READ/DATA.

- Cómo las variables que hay en READ deben ser del mismo tipo –numéricas o alfanuméricas– que las constantes que hay en DATA.
- Cómo se produce un mensaje de error cuando constantes y variables no son del mismo tipo.
- Cómo una sentencia READ puede acceder a una y a varias sentencias DATA, y cómo varias sentencias READ pueden acceder a una sola sentencia DATA.

* La sentencia READ/DATA sólo debe practicarse cuando se ha asimilado perfectamente el contenido de los capítulos anteriores.

EJERCICIOS:

1. Abreviar los siguientes programas utilizando las sentencias READ y DATA:

1.1.
```
10 LET P = 10
20 LET N = 25
30 LET I = P * N
40 LET A$ = "El importe de su conferencia es "
50 PRINT A$; I
60 STOP
```

1.2.
```
 10 LET A = 3
 20 LET B = 5
 30 LET C = 7
 40 LET D = 4
 50 LET E = 2
 60 LET S = A * B + C - D - E
 70 PRINT S
 80 PRINT
 90 LET T = (A + B) * (C - D)
100 PRINT T
110 PRINT
120 LET U = A + (B * C) - (D - E)
130 PRINT U
140 STOP
```

2. Utilizar READ/DATA para escribir programas que, una vez ejecutados, impriman en pantalla:

2.1. El resultado de ordenar de menor a mayor diez números cualesquiera dados en desorden (*)

2.2. El resultado de ordenar de mayor a menor diez números cualesquiera dados en desorden (*).

2.3. El subconjunto P de los números pares menores que 10 y su parte complementaria (números impares menores que 10), dado que el conjunto N de los números naturales menores que 10 (*).

2.4. Dado el conjunto C de camisas que tiene Ana (azul, roja, amarilla y verde) y el conjunto P de pantalones (blanco, azul y verde), hallar el conjunto U de las prendas de vestir que tiene Ana y el conjunto X de las combinaciones que puede hacer Ana utilizando un pantalón y una camisa diferentes.

2.5. Dado el conjunto O de órganos del cuerpo humano (boca, nariz, esófago, laringe, tráquea, pulmones, corazón, bronquios, arterias, intestino, venas, estómago), los órganos que pertenecen al aparato digestivo; los que pertenecen al aparato respiratorio, y los que pertenecen al aparato circulatorio (*).

2.6. Dado el conjunto de los ríos españoles R = [Bidasoa, Nervión, Nalón, Navia, Miño, Duero, Tajo, Guadiana, Guadalquivir, Ebro, Jucar, Segura] los que desembocan en el Cantábrico; los que desembocan en el Atlántico, y los que desembocan en el Mediterráneo (*).

3. Utilizando las sentencias READ y DATA, escribir los programas que sirvan para resolver problemas:

   3.1. En una frutería, hay manzanas de 60 ptas. el kg; melocotones de 100 ptas. el kg; peras de 90 ptas. el kg; melocotones de 65 ptas. el kg, y limones de 75 ptas. el kg. Averiguar el importe de las siguientes compras:

   - 3 kg de manzanas y 2 kg de melocotones
   - 4 kg de melón, 2 Kg de peras y 0,5 kg de limones.
   - 2 kg de manzanas, 1 kg de melocotones, 3 kg de peras, 1,5 kg de melón y 0,75 kg de limones.

   3.2. ¿Cuál es el importe total de las compras anteriores?

   3.3. Si antes de las ventas, en la caja de la frutería había 1850 ptas y ahora, después de las ventas, hay 3498,75 ptas. ¿cuánto dinero falta o sobra?

4. Confeccionar sus propios programas utilizando los comandos READ y DATA.

SOLUCIONES

1.1.

```
  1 REM Ficha:" READ / DATA Ej.1.1"
 10 READ P,N,I,A$
 20 PRINT A$,I
 30 STOP
 40 DATA 10,25,P*N," El importe de la conferencia es "
```

1.2.

```
  1 REM Ficha:" READ / DATA " Ej.1.2"
 20 READ A,B,C,D,E,S
 30 PRINT S'
 40 READ T: PRINT T'
 50 READ U: PRINT U: STOP
 60 DATA 3,5,7,4,2,A*B÷C-D-E,(A+B)*(C-D),A+(B*C)-(D-E)
```

2.4.

```
  1 REM Ficha:" READ / DATA "  Ej.2.4 "
 10 REM "COMBINACIONES"
 20 READ C$,P$,A$,G$,M$,V$,B$
 30 LET U$=V$: LET X$=A$: LET X=0: LET U=0
 40 FOR M=0 TO 2: LET Y$=X$
 50 FOR N=0 TO 3: LET Z$=A$
 60 LET X=X+1: PRINT C$+Z$,P$+Y$
 70 LET A$=G$: LET G$=M$: LET M$=V$: LET V$=Z$
 80 NEXT N: PRINT
 90 LET X$=B$: LET B$=U$: LET U$=Y$
100 NEXT M: LET U=M+N
110 PRINT "Prendas: ",M;"+";N;"=";U'
120 PRINT "Combinaciones: ",X
200 DATA "Camisa ","Pantalon "
210 DATA "azul","gris","marron","verde","blanco"
```

3.

```
  1 REM Ficha:" READ / DATA " Ej.3"
 10 PRINT "COMPRAS"
 20 READ A$,A1,A2,A3,B$,B1,B2,B3,C$,C1,C2,C3
 30 READ D$,D1,D2,D3,E$,E1,E2,E3,F$,F1,F2,K$,P$,T$
 40 LET S=0: PRINT "1a compra: "'A2;K$+A$+P$;A1
 50 LET P=A1*A2: PRINT D2;K$+D$+P$;D1
```

```
 60 LET P=P+D1*D2: PRINT T$;P'': GO SUB 200
 70 PRINT "2a compra: "'B2;K$+B$+P$;B1
 80 LET P=P+B1*B2: PRINT C2;K$+C$+P$;C1
 90 LET PP=P+C1*C2: PRINT E2;K$+E$+P$;E1
100 LET P=P+E1*E2: PRINT T$;P'': GO SUB 200
110 PRINT "3a compra: "'A2;K$+A$+P$;A1
120 LET P=A1*A2: PRINT D3;K$+D$+P$;D1:
130 LET P=P+D1*D3: PRINT C3;K$+C$+P$;C1
140 LET P=P+C1*C3: PRINT E3;K$+C$+P$;E1
150 LET P=P+E1*E3: PRINT T$;P': GO SUB 200
160 PRINT 'F$;F1;"+";S;"=";;F1+S
170 PRINT "ARQUEO, PTAS.: ";F2'"LE FALTAN,PTAS.: ";F1+S-F2
180 STOP
200 LET S=S+P: RETURN
300 DATA "Manzanas",60,3,2,"Melones",100,4,5
310 DATA "Peras",90,2,3,"Melocotones",65,2,1
320 DATA "Limones",75,1,2,"Ptas. en fruteria "
330 DATA 1850,3408," kgs."," por Ptas.","TOTAL COMPRA "
```

RESTORE	**RESTORE número de línea**	**VOLVER A ALMACENAR**

INTERPRETACION:

Con esta sentencia se permite comenzar la lectura de las constantes desde la primera del DATA situado en un número de línea.

Si el número de línea no se especifica, la lectura de constantes se iniciará en la primera del primer DATA del programa.

POSIBILIDADES:

Después de que un programa lee una sentencia RESTORE, la primera sentencia READ que ejecute el programa accederá a la primera constante, según el criterio expuesto en INTERPRETACION.

FORMA DE TECLEAR LAS INSTRUCCIONES

Teclear **RESTORE y el número de línea**, seguido de **ENTER**, para volver a leer las constantes que hay en el *DATA de la línea del número dado.*

Teclear **RESTORE**, seguido de **ENTER**, para volver a leer las constantes desde *el primer DATA del programa.*

EJEMPLOS:

PROGRAMA

```
 1Ø FOR X = 1 TO 5
 2Ø READ V: PRINT V
 3Ø RESTORE
 4Ø NEXT X
1ØØ DATA 3
```

COMENTARIOS

La variable V es leída cinco veces en la línea 2Ø aunque tiene una sola constante en el DATA de la línea 1ØØ. Esto es posible gracias al RESTORE de la línea 3Ø, que obliga a comenzar la lectura en el primer valor del primer DATA.

PROGRAMA

```
 1Ø FOR X = 1 TO 1Ø
 2Ø READ V$ : PRINT V$
 3Ø RESTORE
 4Ø NEXT X
1ØØ DATA "caballo", "jinete"
```

COMENTARIOS

La variable V$ es leída diez veces en la línea 2Ø tomando el primer valor del DATA de la línea 1ØØ.

PROGRAMA

```
 1Ø FOR X = 1 TO 5
 2Ø READ V$ : PRINT V$
```

COMENTARIOS

Imprime cinco veces *montar,* primera (en este caso, única) constante del DATA situado en la

PROGRAMA	COMENTARIOS
3Ø RESTORE 9Ø **4Ø NEXT X** **9Ø DATA "montar"** **1ØØ DATA "jinete", "caballo"**	línea 9Ø. Si fuera RESTORE 1ØØ, la impresión en pantalla sería: *montar* *jinete* *jinete* *jinete* *jinete*

DIDACTICA:

* Recuerde a los alumnos:
  - Que cada sentencia DATA es una lista de valores –numéricos y/o alfanuméricos– separados por comas.
  - Que cada sentencia READ es una lista de nombres de variables –numéricas y/o alfanuméricas– separadas por comas.
  - Que cada vez que el ordenador, en la secuencia normal de lectura, llega a la READ de un valor, toma la primera constante del primer DATA; la siguiente vez, toma la segunda; etc., hasta llegar al final de la lista DATA.
  - Que si en una sentencia READ hay más de una variable, éstas van tomando valores sucesivos de todas las constantes contenidas en los DATA y a partir de la última leída en otro READ anterior, si es que lo ha habido.

* Explique a los alumnos:
  - Que utilizando la sentencia RESTORE podemos asignar valores a todas las variables, aunque haya menos constantes en DATA que variables en READ.
  - Que, con la sentencia RESTORE, podemos hacer que el ordenador salte la lectura de las constantes contenidas en las sentencias DATA a cualquiera de dichas sentencias DATA.
  - Que, para conseguirlo, basta emplear la sentencia RESTORE (indicando el número de línea), con lo que las siguientes sentencias READ comenzarán la lectura de sus datos después del número de línea citado.
  - Que si empleamos la sentencia RESTORE sin número de línea, READ comenzará la lectura de sus datos en la primera sentencia DATA. Es decir, si omitimos el número de línea en RESTORE, es como si hubiésemos tecleado el número de la primera línea del programa y, consiguientemente, el puntero de los DATA se pondrá en el primer valor contenido en las constantes.
  - Que el comando RESTORE, por tanto, permite leer los valores de DATA tantas veces como se desee.

* Que los alumnos observen experimentalmente:
  - Cómo, en un programa donde hay más variables en READ que constantes en DATA, podemos asignar los valores de DATA a todas las variables de READ utilizando adecuadamente la sentencia RESTORE.
  - Cómo podemos conseguir leer varias veces una variable utilizando la sentencia RESTORE, aunque sólo tenga una constante en el DATA que forma parte de un *bucle* FOR/NEXT.

EJERCICIOS:

1. Escribir el programa que, una vez ejecutado, imprima en pantalla dos veces los siguientes datos:

   1Ø DATA Ø, 11, 22, 33, 44, 55, 66, 77, 88, 99

2. Escribir el programa que, una vez ejecutado, imprima en pantalla los siguientes valores:

   m = 1; n = 2; x = 1; y = 2; z = 3 (*).

3. Escribir el programa que, una vez ejecutado, imprima en pantalla cuatro veces:

   Ø 1
   2 3
   4 5
   6 (*)

4. Escribir el programa que, una vez ejecutado, imprima en pantalla:

Cuatro veces:	Ø	1
	2	3
	4	
Una vez:	Ø	1
Una vez:	2	3
	4	5
	6	
Una vez:	Ø	1
	2	

5. Escribir el programa que, una vez ejecutado, imprima en pantalla la media de las calificaciones obtenidas por los siguientes alumnos:

   González: 7.5, 4, 5.5, 8, 6
   Fernández: 6, 8, 9.5, 7, 6.5
   Pérez: 8.5, 3, 6.5, 4, 7 (*)

6. Escribir el programa que, una vez ejecutado, imprima en pantalla el valor de las siguientes ventas:
   5 kg de manzanas
   3 kg de plátanos
   2 kg de melocotones y 5 kg de melón
   3 kg de manzanas, 2 kg de melocotones y 4 kg de melón.
   Los precios por kg de cada fruta son:
   manzanas: 60 ptas; plátanos: 80 ptas; melocotones: 90 ptas; melón: 85 ptas. (*)

7. Completar el programa anterior para que, una vez ejecutado, imprima en pantalla las cantidades de fruta que faltan por vender, sabiendo que había:

   15 kg de manzanas; 12 kg de plátanos; 10 kg de melocotones, y 25 kg de melón (*).

8. Completar el programa anterior para que, una vez ejecutado, imprima en pantalla:
   1. El valor total de las ventas.
   2. El valor total de las existencias (*)

9. Escribir el programa que, una vez ejecutado, permita introducir mediante el teclado el nombre de una fruta; que explore la sentencia DATA (nombres de frutas y cantidad en kg de cada fruta), e imprima el nombre de la fruta elegida y la cantidad en existencia.

10. Idear sus propios programas utilizando las sentencias READ, DATA y RESTORE, además de otras conocidas.

1.
```
  1 REM Ficha:" RESTORE " Ej.1"
 10 FOR N=1 TO 2
 20 RESTORE
 30 FOR M=0 TO 9
 40 READ A: PRINT A;" ";
 50 NEXT M
 60 PRINT
 70 NEXT N
100 DATA 0,11,22,33,44,55,66,77,88,99
```

4.
```
  1 REM Ficha:" RESTORE " Ej.4"
 10 LET X=5: LET Y=3: LET Z=0
 20 FOR N=0 TO Y
 30 FOR M=1 TO X
 50 READ A: PRINT A,
 60 NEXT M: IF N<Y THEN  RESTORE
 70 PRINT ': NEXT N: PRINT '': LET Z=Z+1
 80 IF Z=1 THEN  LET Y=0: LET X=2: RESTORE
 90 IF Z=2 THEN  LET X=5
100 IF Z=3 THEN  LET X=3: RESTORE
110 IF Z=4 THEN  STOP
120 GO TO 20
200 DATA 0,1,2,3,4,5,6,2,3,4,5,6
```

9.
```
  1 REM Ficha:" RESTORE " Ej.9"
  5 PRINT "Escriba el nombre de una fruta. la inicial
    en mayusculas y en   plural.": REM "BUSQUEDA"
 10 PRINT : INPUT A$
 20 FOR N=1 TO 11
 30 READ B$,C
 40 IF A$=B$ THEN  PRINT B$,C;" KILOS": RESTORE : GO
    TO 10
 50 NEXT N
 60 PRINT "No existen "+A$: RESTORE : GO TO 10
200 DATA "Naranjas",105,"Cerezas",43,"Peras",87,"Plat
    anos",36,"Uvas",112,",Higos",99,"Ciruelas",54,"Mel
    ocotones",17,"Pomelos",75,"Almendras",166,"Manzana
    s",88
```

INT	INT (expresión numérica)	ENTERO

INTERPRETACION:

Se obtiene el mayor de los números enteros menores –o igual– que el indicado en la expresión numérica.

POSIBILIDADES:

El valor de la expresión numérica puede ser cualquier número real.

FORMA DE TECLEAR LA INSTRUCCION

Teclear **INT y la expresión numérica entre paréntesis**, seguido de **ENTER.**

EJEMPLOS:

PROGRAMA 1	COMENTARIOS
1Ø LET x = 89.98 **2Ø PRINT INT x**	Al ejecutar este programa, obtendremos 89 impreso en la pantalla, que es el mayor número entero menor que 89.98.

PROGRAMA 2	COMENTARIOS
1Ø LET x = –10.17 **2Ø PRINT INT x**	Al ejecutar este programa, obtendremos –11 impreso en la pantalla, que es el mayor número menor que –10.17.

PROGRAMA 3	COMENTARIOS
1Ø LET x = 10.17 **2Ø PRINT INT x**	En este caso, obtendremos 1Ø impreso en la pantalla, que es el mayor número entero menor que 10.17.

DIDACTICA:

* Recuerde a los alumnos:

- Que, en Informática, no se pone punto para separar las unidades de millar: los números enteros se escriben sin separación alguna.
- Que, en cambio, en los números decimales se utiliza el punto para separar la parte entera de la decimal.

- Que de dos números positivos, es menor el que tiene menor valor absoluto. Por ejemplo, 9 < 10.
- Que de dos números negativos, es menor el que tiene mayor valor absoluto. Por ejemplo, −10 < −9.

* Explique a los alumnos:
  - Que la función INT suprime los decimales de la expresión numérica dejando solamente la parte entera.
  - Que con la función INT no se redondea dicha expresión numérica, sino que se obtiene el mayor de los números enteros (positivo o negativo) menores o igual que la misma.
  - Que la función INT es muy útil en los programas de cálculo, de juegos y otros más complejos.

* Que los alumnos observen experimentalmente:
  - Cómo la función INT suprime los decimales en las expresiones numéricas —positivas y negativas— y qué valores toman.
  - Cómo puede utilizarse la función INT para redondear una expresión numérica decimal al entero más próximo sumando 0.5 a su valor.

EJERCICIOS:

1. Ejecutar el siguiente programa:

```
10 LET X = 4.57
20 PRINT INT X
```

2. Modificar el programa anterior dando a **X** otros valores de su elección (*).

3. Escribir los programas que, una vez ejecutados, impriman en pantalla la parte entera del argumento numérico de las siguientes expresiones numéricas:

```
 3.425
-3.425
 6.25 + 4.37
-6.25 - 4.37
 6.25 - 4.37
-6.25 + 4.37
 5.8/2.3
 5.8/-2.3
-5.8/2.3
-5.8/-2.3
 7.45 * 4.20
-7.45 * 4.20
 7.45 * (-4.20)
 (-7.45) * (-4.20)
```

4. Explicar los resultados obtenidos.

5. Redondear al entero más próximo el valor de las anteriores expresiones numéricas.

3.

```
  1 REM Ficha:"INT " Ej.3"
  5 PRINT "PARTE ENTERA DE UN NUMERO"''
 10 PRINT 3.425,INT (3.425)
 20 PRINT -3.425,INT (-3.425)
 30 PRINT 6.25+4.37,INT (6.25+4.37)
 40 PRINT -6.25+-4.37,INT (-6.25-4.37)
 50 PRINT 6.25-4.37,INT 6.25-4.37
 60 PRINT -6.25+4.37,INT (-6.25+4.37)
 70 PRINT 5.8/2.3,INT (5.8/2.3)
 80 PRINT 5.8/-2.3,INT (5.8/-2.3)
 90 PRINT -5.8/2.3,INT (-5.8/2.3)
100 PRINT -5.8/-2.3,INT (-5.8/-2.3)
110 PRINT 7.45*4.2,INT (7.45*4.20)
120 PRINT -7.45*4.20,INT (-7.45*4.20)
130 PRINT 7.45*(-4.20),INT (7.45*(-4.20))
140 PRINT (-7.45)*(-4.20),INT ((-7.45)*(-4.20))
```

5.

```
  1 REM Ficha:"INT " Ej.5"
  5 PRINT "PARTE ENTERA DE UN NUMERO"'"REDONDEADA AL
    ENTERO PROXIMO"''
 10 PRINT 3.425,INT (3.425+.5)
 20 PRINT -3.425,INT (-3.425+.5)
 30 PRINT 6.25+4.37,INT (6.25+4.37+.5)
 40 PRINT -6.25+-4.37,INT (-6.25-4.37+.5)
 50 PRINT 6.25-4.37,INT (6.25-4.37+.5)
 60 PRINT -6.25+4.37,INT (-6.25+4.37+.5)
 70 PRINT 5.8/2.3,INT (5.8/2.3+.5)
 80 PRINT 5.8/-2.3,INT (5.8/-2.3+.5)
 90 PRINT -5.8/2.3,INT (-5.8/2.3+.5)
100 PRINT -5.8/-2.3,INT (-5.8/-2.3+.5)
110 PRINT 7.45*4.2,INT (7.45*4.20+.5)
120 PRINT -7.45*4.20,INT (-7.45*4.20+.5)
130 PRINT 7.45*(-4.20),INT (7.45*(-4.20)+.5)
140 PRINT (-7.45)*(-4.20),INT ((-7.45)*(-4.20)+.5)
```

RND/RANDOMIZE		ALEATORIO/ALEATORIZAR

RND	RND	ALEATORIO

INTERPRETACION:

Cada vez que se utiliza RND, se obtiene un valor diferente y aleatorio igual o mayor que Ø y menor que 1.

POSIBILIDADES:

Esta función da la misma secuencia de valores cada vez que se ejecuta el programa que la contiene, tecleando RUN/ENTER.

FORMA DE TECLEAR LA INSTRUCCION

Teclear **RND**, seguido de **ENTER.**

EJEMPLOS:

1. Teclee en forma directa estas tres instrucciones y observe los resultados:

   PRINT RND y pulse ENTER
   PRINT RND y pulse ENTER
   PRINT RND y pulse ENTER

2. Teclee este programa en su computador y córralo varias veces, comparando los resultados:

```
10 FOR X = 0 TO 20
20 PRINT RND
30 NEXT X
```

3. Teclee en forma directa estas instrucciones y observe los resultados:

   PRINT RND * 1Ø y pulse **ENTER**
   PRINT RND * (5 + 2) y pulse **ENTER**
   PRINT (5 + 2) * RND y pulse **ENTER**
   PRINT (1.3 + Ø.7) * RND y pulse **ENTER**
   PRINT RND * (1.3 + Ø.7) y pulse **ENTER**
   PRINT (−1Ø) * RND y pulse **ENTER**

RANDOMIZE	RANDOMIZE expresión numérica	ALEATORIZAR

INTERPRETACION:

Inicializa el generador de números aleatorios de acuerdo con la expresión numérica, con lo que la secuencia de números RND comenzará con valores distintos.

POSIBILIDADES:

El valor de la expresión numérica debe estar comprendido entre 1 y 65535, con lo que las secuencias RND comenzarán en diferentes lugares.

En algunos BASIC, el valor de expresión numérica debe estar comprendido entre –32768 y 32767.

RANDOMIZE Ø obliga a RND a producir números lo más aleatorios posible.

En algunos desarrollos técnicos y/o científicos, convendrá simular fenómenos aleatorios controlados, de forma tal que, variando algunos parámetros, se repita el proceso con los mismos valores.

FORMA DE TECLEAR LA INSTRUCCION

Teclear **RANDOMIZE y la expresión numérica**, seguido de **ENTER.**

EJEMPLO:

1. Escriba un programa que produzca números lo más aleatorios posible, independientemente de que se ejecute o se inicialice.

SOLUCION:

PROGRAMA	COMENTARIOS
1Ø RANDOMIZE Ø **2Ø FOR X = 1 TO 2Ø** **3Ø PRINT RND** **4Ø NEXT X**	Córralo varias veces y comprobará que, a diferencia del segundo ejemplo expuesto en la ficha de RND, los resultados son diferentes en cada ocasión.

EJEMPLO:

2. Basándose en una misma secuencia de números aleatorios, obtenga las series de resultados que se deriven de los diferentes valores que se den por teclado.

SOLUCION:

PROGRAMA	COMENTARIOS
1Ø INPUT A	Entrada de valores.
15 RANDOMIZE 1	Se fija el punto de partida del generador de números aleatorios.
2Ø FOR X = 1 TO 2Ø **3Ø PRINT RND * A** **4Ø NEXT X** **5Ø GO TO 1Ø**	El resto del programa está dedicado al proceso en estudio que, en este caso, es una simple multiplicación del valor dado en la línea 1Ø por el RND correspondiente.

DIDACTICA:

* Recuerde a los alumnos:
  - Que la función INT suprime los decimales dejando solamente el mayor de los números enteros menores o igual que la expresión numérica.
  - Qué es el azar o casualidad y cómo interviene en los sucesos de la vida real y en los juegos.

* Explique a los alumnos:
  - Que con la función RND podemos introducir el azar en un programa para simular sucesos.
  - Que, cada vez que se introduce RND, se obtiene un valor diferente y aleatorio igual o mayor que Ø y menor que 1.
  - Que con RND se pueden obtener números aleatorios en intervalos diferentes entre Ø y 1. Es decir, que para obtener intervalos:

    entre Ø y < 1 hay que pulsar RND
    entre Ø y < n hay que pulsar (n) * RND
    entre n y < m hay que pulsar (n–m) * RND

  - Que para obtener números enteros, debemos utilizar INT (n + m * RND).
  - Que la función RND no se expresa igual en todos los ordenadores, por lo que debe consultarse detenidamente el MANUAL de cada ordenador.
  - Que la sentencia RANDOMIZE se utiliza para hacer que la función RND comience su secuencia de números en un lugar determinado.
  - Que se pueden utilizar números entre 1 y 65535 en la sentencia RANDOMIZE para comenzar la secuencia RND en diferentes lugares.
  - Que RANDOMIZE Ø obliga a RND a producir números lo más aleatorios posible.

* Sólo para profesores: La función RND no es verdadera aleatoria, ya que sigue una secuencia fija de 65536 números.
  Sin embargo, al ser tan complicada esta secuencia, es casi imposible tener pautas previas para conocerla, por lo que se puede considerar RND como pseudoaleatoria.

* Las sentencias RND y RANDOMIZE deben ser tratadas paulatinamente y adaptándolas a la mentalidad de los alumnos. Lo más importante es que comprendan el concepto de aleatoriedad que se obtiene con RND y la posibilidad de comenzarla en diferentes números, gracias a RANDOMIZE.

* Que los alumnos observen experimentalmente:

- Cómo RND obtiene un valor diferente y aleatorio igual o mayor que Ø y menor que 1 cada vez que se utiliza.
  Pueden obtener diferentes valores tecleando en forma directa varias instrucciones PRINT RND/ENTER.
- Cómo pueden obtener números aleatorios en intervalos diferentes desde un valor igual o mayor que Ø y menor que un número **n.**
- Cómo pueden obtener números aleatorios en intervalos diferentes desde un valor igual o mayor que un número **n** y menor que otro número **m.**
- Cómo poder obtener números enteros utilizando INT (RND).
- Cómo se utiliza la secuencia RANDOMIZE para hacer que la función RND comience su secuencia de números en un lugar determinado.
- Cómo utilizar un bucle FOR/NEXT para obtener una secuencia de números aleatorios.
- Cómo obtener una serie de resultados derivados de los diferentes valores dados por teclado, basándose en una misma secuencia de números aleatorios.

EJERCICIOS:

Escribir los programas que, una vez ejecutados, impriman en pantalla:

1. Cien números aleatorios iguales o mayores que Ø y menores que 1.
2. Cien números aleatorios iguales o mayores que Ø y menores que 1Ø.
3. Cien números aleatorios iguales o mayores que 5 y menores que 15.
4. Cien números aleatorios iguales o mayores que 125Ø (*).
5. El número de veces que ha salido "cara" y el número de veces que ha salido "cruz" al lanzar cincuenta veces una moneda al aire.
6. Los resultados de lanzar cien veces un dado
7. El número de veces que han salido seis puntos en cien tiradas de dado (*).
8. El número de veces que ha salido cada cara del dado en cien tiradas (*).
9. Utilizando tres dados de *poker*, los resultados obtenidos en cincuenta tiradas (*).
10. El número de veces que han salido tres ases en cincuenta tiradas con los cinco dados de *poker* (*).
11. Idear sus propios programas utilizando INT, RND y RANDOMIZE, además de otros comandos conocidos.

SOLUCIONES

1.
```
  1 REM Ficha:"RND/ RANDOMIZE " Ej.1"
 10 FOR n=1 TO 100
 20 PRINT RND
 30 NEXT n
```

2.
```
  1 REM Ficha:"RND/ RANDOMIZE " Ej.2"
 10 FOR n=1 TO 100
 20 PRINT 10*RND
 30 NEXT n
```

3.
```
  1 REM Ficha:"RND/ RANDOMIZE " Ej.3"
 10 FOR n=1 TO 100
 20 PRINT 10*RND+5
 30 NEXT n
```

5.
```
  1 REM Ficha:"RND/ RANDOMIZE " Ej.5"
  5 PRINT "MONEDA AL AIRE"''
 10 RANDOMIZE : LET B=0: LET c=0: LET d=0: FOR N=1 TO 50
 20 LET A=100*RND: LET B=B+a
 30 IF B/50>25 THEN  LET C=C+1
 40 IF B/50<25 THEN  LET D=D+1
 50 NEXT N
 60 PRINT C;" veces CARA",
 70 PRINT d;" veces CRUZ"
100 PAUSE 0: GO TO 10
```

6.
```
  1 REM Ficha:"RND/ RANDOMIZE " Ej.6"
  5 PRINT "TIRADA DE DADOS"
 10 LET B=0: LET C1=0: LET C2=0: LET C3=0: LET C4=0: L
    ET C5=0: LET C6=0
 20 RANDOMIZE : FOR N=1 TO 100
 30 LET B=601*RND
 40 IF B/6>=83.33 THEN  LET C1=C1+1
 50 IF B/6>=66.67 THEN  IF B/6<83.33 THEN  LET C2=C2+1
 60 IF B/6>=50 THEN  IF B/6<66.67 THEN  LET C3=C3+1
 70 IF B/6>=33.33 THEN  IF B/6<50 THEN  LET C4=C4+1
 80 IF B/6>=16.67 THEN  IF B/6<33.33 THEN  LET C5=C5+1
 90 IF B/6<16.67 THEN  LET C6=C6+1
100 NEXT N
110 PRINT C1;" veces 1",C2;" veces 2"
120 PRINT C3;" veces 3",C4;" veces 4"
130 PRINT C5;" veces 5",C6;" veces 6"''
140 PAUSE 0: GO TO 10
```

SQR	SQR (expresión numérica)	RAIZ CUADRADA

INTERPRETACION:

Se obtiene el valor de la raíz cuadrada de la expresión numérica.

POSIBILIDADES:

El valor de la expresión numérica debe ser siempre igual o mayor que Ø. Por razones matemáticas, hay que tener en cuenta que **SQR** (expresión numérica) = (expresión numérica) elevada a (1/2).

FORMA DE TECLEAR LA INSTRUCCION

Teclear **SQR y la expresión numérica entre paréntesis,** seguido de **ENTER.**

EJEMPLO:

1. Teclee en forma directa:

   PRINT SQR 25 y pulse ENTER

   Obtendrá como respuesta del computador 5, que es el valor de la raíz cuadrada de 25:

   $\sqrt{25} = 25 \uparrow (1/2) = 5$

EJEMPLO:

2. Determine, mediante un programa, las raíces cuadradas de los números enteros comprendidos entre Ø y 2Ø.

SOLUCION:

PROGRAMA	COMENTARIOS
1Ø FOR X = Ø TO 2Ø **2Ø PRINT "Raíz cuadrada de "; X; "="; SQR X** **3Ø NEXT X**	Corra este programa tecleando RUN/ENTER y observe la impresión en pantalla.

SGN	SGN (expresión numérica)	SIGNO

INTERPRETACION:

Esta función sólo nos devolverá 1, ∅ ó −1, según el valor de la expresión numérica sea mayor, igual o menor que ∅, respectivamente.

POSIBILIDADES:

Si la expresión numérica es mayor que ∅, SGN (expresión numérica) nos dará el valor 1.

Si la expresión numérica es igual que ∅, SGN (expresión numérica) nos dará el valor ∅.

Si la expresión numérica es menor que ∅, SGN (expresión numérica) nos dará el valor −1.

FORMA DE TECLEAR LA INSTRUCCION

Teclear **SGN y la expresión numérica entre paréntesis**, seguido de **ENTER.**

EJEMPLO:

1. Teclee en forma directa:

PRINT SGN (4−206) y pulse ENTER
Obtendrá como respuesta del computador: −1.

EJEMPLO:

2. Interprete estas líneas:

```
1∅ INPUT A
2∅ INPUT B
3∅ IF SGN (A − B) = 1 THEN GO TO 1∅∅∅
4∅ IF SGN (A − B) = − 1 THEN GO TO 2∅∅∅
5∅ PRINT "Valores iguales, repita, por favor" : GO TO
................
................
```

SOLUCION:

En las líneas 1∅ y 2∅ se introducen por teclado dos valores.
En la línea 3∅, saltará a la línea 1∅∅∅ si SGN (A−B) es 1.
En la línea 4∅, saltará a la línes 2∅∅∅ si SGN (A−B) es −1.
En otro caso −SGN (A−B) x ∅−, se emite un mensaje −"Valores iguales, repita, por favor"− y se pasa a la línea 1∅.

INTERPRETACION:

Se obtiene el valor absoluto de la expresión numérica.

POSIBILIDADES:

La expresión numérica puede ser un número o una expresión numérica que, como ya es sabido, finalmente es un número.

FORMA DE TECLEAR LA INSTRUCCION

Teclear **ABS y la expresión numérica entre paréntesis**, seguido de **ENTER.**

EJEMPLO:

PROGRAMA	COMENTARIOS
PRINT ABS (–5)	Si tecleamos esta instrucción y apretamos ENTER obtenemos el valor absoluto de –5, que es 5.

> Cuando se teclea una instrucción sin línea de programa, esperando respuesta inmediata del ordenador, estamos trabajando en modo inmediato.

EJEMPLO:

PROGRAMA	COMENTARIOS
1Ø LET a = 25 **2Ø LET b = 1Ø** **3Ø PRINT ABS (25 * 1Ø)**	En este caso, la respuesta del ordenador será 25Ø.

LN o LOG	LN (expresión numérica)	LOGARITMO

INTERPRETACION:

Se obtiene el valor del logaritmo natural de la expresión numérica.

POSIBILIDADES:

En algunos *dialectos* del BASIC, es posible encontrar **LN** como **LOG**, siendo su función la misma.

El valor de la expresión numérica debe ser mayor que Ø.

FORMA DE TECLEAR LA INSTRUCCION

Teclear **LN** o **LOG y la expresión numérica entre paréntesis,** seguido de **ENTER.**

EJEMPLO:

Obtenga el logaritmo en base **e** de 6.5, utilizando su ordenador en forma directa.

SOLUCION:

PRINT LN 6.5

Pulse ENTER y obtendrá la respuesta.

EXP	EXP (expresión numérica)	EXPONENCIAR e

INTERPRETACION:

Se obtiene el valor del número **e** elevado a la potencia representada por la expresión numérica.

FORMA DE TECLEAR LA INSTRUCCION

Teclear **EXP y la expresión numérica entre paréntesis,** seguido de **ENTER.**

EJEMPLO:

Elevar el número **e** a la segunda potencia.

SOLUCION:

Teclee **PRINT EXP (2)** y pulse **ENTER**. La pantalla le mostrará el resultado.

INTERPRETACION:

Se obtiene el valor del seno de la expresión numérica, dado el ángulo en radianes.

POSIBILIDADES:

Las que se derivan del cálculo trigonométrico.

FORMA DE TECLEAR LA INSTRUCCION

Teclear **SIN y la expresión numérica entre paréntesis,** seguido de **ENTER.**

EJEMPLO:

Escribir un programa que permita calcular el seno de un ángulo dado en grados sexagesimales.

SOLUCION:

360° < > 2 π. Esto quiere decir que si **n** es el ángulo en grados sexagesimales, los radianes equivalentes son:

$$n \frac{2\pi}{360}$$

PROGRAMA	COMENTARIOS
1Ø INPUT "Grados sexagesimales"; n **2Ø LET X = n * 2 * PI/360** **3Ø PRINT n; " ("; X; ")", SIN X**	Una vez realizada la transformación a radianes (línea 2Ø), se imprime el valor del ángulo en grados sexagesimales; a continuación, entre paréntesis, su equivalente en radianes, y, después, el valor del seno.

INTERPRETACION:

Se obtiene el valor del coseno de la expresión numérica, dado el ángulo en radianes.

POSIBILIDADES:

Las que se derivan del cálculo trigonométrico.

FORMA DE TECLEAR LA INSTRUCCION

Teclear **COS y la expresión numérica entre paréntesis**, seguido de **ENTER.**

EJEMPLO:

Escribir el programa que, al ejecutarlo, imprima en pantalla el valor de los cosenos de los ángulos de 0,5 en 0,5 radianes, entre 0 y 2 π.

SOLUCION:

PROGRAMA	COMENTARIOS
1Ø FOR X = Ø TO 2 * PI STEP 0.5 **2Ø PRINT COS X** **3Ø NEXT X**	Si el computador en uso dispone de otras funciones trigonométricas, los criterios de utilización de las mismas serán similares a los expuestos hasta aquí.

DEF FN	DEF FN variable = expresión	DEFINIR FUNCION

INTERPRETACION:

Con esta instrucción, el usuario del ordenador puede definir sus propias funciones, siguiendo las reglas que se explican a continuación y de forma análoga al proceso mental matemático de decir: "**La función y de x – y (x) – es igual a tal expresión**".Estas reglas varían de un ordenador a otro, especialmente en el sentido de afectar o no a las variables utilizadas en el programa, en el caso de tener los mismos *nombres* que las de la **variable** y las contenidas en la **expresión.**

POSIBILIDADES:

Las funciones numéricas se reconocen y se denominan mediante instrucciones **FN** seguidas de una sola letra. Por ejemplo, **DEF FN A.**

Según el tipo de ordenador, en la variable pueden referenciarse una o más variables.

FORMA DE TECLEAR LA INSTRUCCION

Teclear **DEF FN, el nombre de la variable, el signo igual (=) y la expresión numérica**, seguido de **ENTER.**

EJEMPLO:

Para definir una función en BASIC se debe proceder de una forma similar a ésta:

1Ø DEF FN A (x) = x + x * 2

La función aquí definida nos dice que, para un valor de **x**, la función **A (x)** es igual a la expresión indicada. Dicho esto, cabe preguntarse cómo se dan valores a la variable **x**. Lo veremos a continuación al tratar **el comando FN.**

FN	FN variable valor de la variable	FUNCION

INTERPRETACION:

Esta instrucción es el complemento imprescindible de la **DEF FN,** ya que con ella se calcula la *función definida* de acuerdo con el valor dado a la variable involucrada.

POSIBILIDADES:

En trabajos matemáticos o técnicos, estos comandos son muy prácticos. Por esta razón, hemos hecho aquí una referencia a los mismos, pero su campo de posibilidades depende de lo que permita hacer con ellos cada ordenador.

FORMA DE TECLEAR LA INSTRUCCION

Teclear **FN, el nombre de la variable y el valor de la variable entre paréntesis,** seguido de **ENTER.**

EJEMPLO:

En el caso propuesto en el ejemplo de **DEF FN,** el **FN** funcionaría así para una expresión numérica:

SOLUCION:

PROGRAMA	COMENTARIOS
1∅ DEF FN A (x) = x + x * 2 **2∅ LET y = FN A (7)** **3∅ PRINT y**	En la línea 1∅ se define la función. En la línea 2∅ se da el valor 7 a la variable x y se calcula la expresión definida en 1∅ con este valor.

DIDACTICA:

* Recuerde a los alumnos:
  - Que un ordenador puede realizar cálculos matemáticos a gran velocidad.
  - Los símbolos (*operadores aritméticos*) que sirven para hacer algunos de estos cálculos:

+	–	*	/	↑ o ∧
sumar	restar	multiplicar	dividir	exponenciar

  - El orden de prioridad en que el ordenador realiza estas operaciones y, en consecuencia, las reglas de su utilización.

* Explique a los alumnos:

– Que, además de los operadores aritméticos anteriores, el BASIC dispone de otras funciones incorporadas al ordenador para facilitar la programación, tales como **INT** y **RND/RANDOMIZE**, ya conocidas.

– Para qué sirven las funciones incorporadas:

SQR	Para calcular la raíz cuadrada de una expresión numérica.
SGN	Para conocer el signo de una expresión numérica.
ABS	Para obtener el valor absoluto de una expresión numérica.
LN (LOG)	Para hallar el valor del logaritmo natural de una expresión numérica.
EXP	Para obtener el valor del número **e** elevado a una expresión numérica.
SIN	Para calcular el seno de una expresión numérica.
COS	Para calcular el coseno de una expresión numérica.
TAN	Para calcular la tangente de una expresión numérica.

– Que si quiere utilizar una función que no está incorporada al ordenador, puede definirla utilizando las instrucciones **DEF FN** y su complementaria **FN**, siempre teniendo en cuenta las limitaciones que imponga el ordenador en uso.

* Que los alumnos observen experimentalmente:

– Cómo se realizan cálculos con los operadores aritméticos.

– Cómo se obtienen la raíz cuadrada, el signo de una expresión numérica, el valor absoluto, etc. utilizando los comandos explicados.

– Cómo se obtiene la raíz **n** de una expresión numérica.

Recuerde a sus alumnos que, por razones matemáticas, la raíz **n** de cualquier expresión numérica es igual a esa expresión numérica elevada a **1/n**. Por ejemplo, $\sqrt[n]{x} = x^{1/n}$

– Cómo:

Si una expresión numérica es mayor que cero

SGN (expresión numérica) = 1

Si una expresión numérica es igual que cero

SGN (expresión numérica) = Ø

Si una expresión numérica es menor que cero

SGN (expresión numérica) = -1

– Que el valor absoluto de una expresión numérica positiva o negativa es siempre un número positivo.

– Que para poder hallar el logaritmo natural de una expresión numérica, el valor de ésta debe ser mayor que cero.

– Que el valor del número **e** elevado a una expresión numérica es el antilogaritmo de esa expresión numérica.

– Que para calcular las funciones trigonométricas de una expresión numérica el ángulo debe estar expresado en radianes:

$$n \frac{2\pi}{360}$$

– Cómo se pueden definir otras funciones, utilizando los comandos **DEF FN** y **FN**, con expresiones numéricas y alfanuméricas.

EJERCICIOS:

El profesor debe proponer los que estime oportunos para utilizar los comandos estudiados, naturalmente, dependiendo de los conocimientos matemáticos de sus alumnos. Por ejemplo:

SOLUCIONES

```
  1 REM Ficha:"SQR " Ej.1"
 10 CLS : PRINT AT 0,0;"ECUACION DE 2o.GRADO"''
 20 INPUT "VALOR DE a: ";a: PRINT "a= ";A''
 30 INPUT "VALOR DE b: ";B: PRINT "b= ";B''
 40 INPUT "VALOR DE c: ";C: PRINT "c= ";C
 45 PRINT AT 10,0;"Ecuacion: ";a;"x^2";
 47 IF B>0 THEN  PRINT "+";b;"X";
 49 IF B<0 THEN  PRINT b;"X";
 51 IF c>0 THEN  PRINT "+";c;
 53 IF c<0 THEN  PRINT c;
 55 PRINT "= 0"''
 57 IF ((b*b)-(4*a*c))<0 THEN  PRINT ''"Raiz imaginaria":
    GO TO 100
 58 IF 2*a=0 THEN  PRINT "X' = 0/0","X''= 0/0": GO TO 100
 60 LET x1=(-b+(SQR ((b*b)-(4*a*c))))/(2*a)
 70 LET x2=(-b-(SQR ((b*b)-(4*a*c))))/(2*a)
 80 PRINT "X' = ";X1''
 90 PRINT "X''= ";X2
100 PAUSE 0: GO TO 10

  1 REM Ficha:"LN "
 10 PRINT "LOGARITMOS VULGARES"''
 20 INPUT "Numero ";N
 30 PRINT " Log.de ";n;" = ",LN N/LN 10
 40 GO TO 20

  1 REM Ficha:"EXP "
 10 PRINT "ANTILOGARITMOS VULGARES"''
 20 INPUT "Logaritmo: ";L
 30 PRINT "Antlog.de ";L;" = ";EXP (L*LN 10)
 40 GO TO 20
```

INTERPRETACION:

Se obtiene el carácter cuyo código es la expresión numérica indicada.

POSIBILIDADES:

Los valores a asignar en la expresión numérica deben estar comprendidos entre Ø y 255, ambos inclusive, que corresponden **al código ASCII de caracteres.**

Una introducción al **código ASCII** se estudia en otro apartado de esta publicación.

FORMA DE TECLEAR LA INSTRUCCION

Teclear **CHR$ y la expresión numérica entre paréntesis,** seguido de **ENTER.**

EJEMPLO:

CHR$ 65 corresponde al carácter **A.** Para comprobarlo, basta teclear PRINT CHR$ 65 seguido de ENTER.

CODE o ASC	CODE (expresión alfanumérica)	CODIGO

INTERPRETACION:

Se obtiene el código ASCII del primer carácter de la expresión alfanumérica.

POSIBILIDADES:

Es la función inversa de CHR$. La mayoría de los ordenadores utiliza ASC en lugar de CODE.

FORMA DE TECLEAR LA INSTRUCCION

Teclear **CODE** o **ASC y la expresión alfanumérica entre paréntesis**, seguido de **ENTER.**

EJEMPLO:

CODE "Angel" nos retornará **65**, que es el código de A, primer carácter de la expresión alfanumérica "Angel". Para comprobarlo, basta teclear PRINT CODE "Angel" seguido de ENTER.

LEN	LEN (expresión alfanumérica)	LONGITUD

INTERPRETACION:

Se obtiene el número de caracteres que componen la expresión alfanumérica, incluyendo los espacios.

POSIBILIDADES:

El resultado que se obtiene con LEN es, a todos los efectos, un número y, por tanto, podemos utilizarlo como tal según convenga.

También podemos usar LEN en la suma de cadenas o de variables de caracteres.

FORMA DE TECLEAR LA INSTRUCCION

Teclear **LEN y la expresión alfanumérica entre paréntesis**, seguido de **ENTER.**

EJEMPLO:

Sumar 1Ø al número de caracteres que contenga la suma de la variable **a$** y la cadena "**ABC**", suponiendo que **a$** = "**ZXY**".

SOLUCION:

PROGRAMA	COMENTARIOS
1Ø LET a$ = "ZXY"	En la primera línea, asignamos a la variable de caracteres **a$** la cadena indicada "**ZXY**".
2Ø LET a = LEN (a$ + "ABC")	En la línea 2Ø, asignamos a la variable numérica **a** el número correspondiente al total de caracteres existente entre la cadena representada por a$ y la "**ABC**".
3Ø PRINT a + 1Ø	En la última línea, ordenamos la impresión del resultado de la suma de la variable **a** más 1Ø.

STR$	STR$ (expresión numérica)	CADENA

INTERPRETACION:

Se obtiene una cadena de caracteres compuesta por los mismos dígitos que componen la expresión numérica y en el mismo orden.

POSIBILIDADES:

Por ser el resultado obtenido una cadena de caracteres, no podrá ser manejado como el número que era en la expresión numérica, pero sí —a todos los efectos— como la cadena que es.

FORMA DE TECLEAR LA INSTRUCCION

Teclear **STR$ y la expresión numérica entre paréntesis**, seguido de **ENTER.**

EJEMPLO:

PROGRAMA	COMENTARIOS
1Ø LET a = 1984 **2Ø LET b$ = STR$ a** **3Ø PRINT a, b$**	En la línea 2Ø, asignamos a la variable de caracteres **b$** la cadena equivalente al número contenido en la variable **a**. Cuando ejecute este programa, observará que, en la misma línea y separadas de acuerdo con la coma existente entre **a** y **b$**, aparece dos veces 1984. Bien, pues la primera es a todos los efectos un número, ya que procede del valor contenido en la variable numérica **a**, mientras que la segunda es una cadena de caracteres.

VAL	VAL (expresión alfanumérica)	VALOR

INTERPRETACION:

Se obtiene el valor numérico de la expresión alfanumérica indicada.

POSIBILIDADES:

Es necesario advertir que para poder aplicar la función VAL sobre una expresión alfanumérica ésta debe estar compuesta exclusivamente por dígitos (números del Ø al 9), sin ningún signo entre ellos.

FORMA DE TECLEAR LA INSTRUCCION

Teclear **VAL y la expresión alfanumérica entre paréntesis**, seguido de **ENTER.**

EJEMPLO:

PROGRAMA	COMENTARIOS
1Ø LET a$ = "1984" **2Ø PRINT VAL (a$) + 2**	La respuesta a este programa es 1986, suma del valor de la variable a$ –transformada– más 2.

INKEY$	INKEY$	APRETAR TECLA

INTERPRETACION:

Se obtiene el carácter de la tecla que se aprieta o la cadena vacía, en el momento en que se ejecuta la sentencia INKEY$. Dicho de otro modo, si al llegar la lectura del programa a la función INKEY$ hay alguna tecla apretada, entonces se obtendrá el carácter de dicha letra; en caso contrario, el computador la considerará cadena vacía.

> Se entiende por **cadena vacía** aquélla que no contiene ningún carácter. Se representa por dos comillas consecutivas (" "), de tal modo que si escribimos, por ejemplo, **LET A$** = " ", el computador considerará que la variable de caracteres A$ no contiene ninguno.

POSIBILIDADES:

Esta función se puede aplicar directamente o a través de una variable de caracteres. Esto quiere decir que INKEY$, al ser una función y, por tanto, devolver un valor —un carácter, en este caso—, puede ser manejada como una cadena independiente, haciéndola igual a una variable alfanumérica o ella por sí misma.

FORMA DE TECLEAR LA INSTRUCCION

Teclear **INKEY$**, seguido de **ENTER**.

EJEMPLO:

Diseñar un bucle, dentro de un hipotético programa, que impida seguir la lectura del mismo hasta que se pulse una tecla cualquiera.

SOLUCION:

PROGRAMA	COMENTARIOS
1Ø .... **3Ø IF INKEY$ = " " THEN GO TO 3Ø** 5Ø	En función de lo expuesto anteriormente, en la línea 3Ø imponemos la condición de que mientras INKEY$ dé como resultado la cadena vacía, la lectura del programa vaya a su propia línea. Es decir, mientras no se apriete una tecla cualquiera, el programa quedará detenido en la línea donde está INKEY$.

EJEMPLO:

Escribir un programa que, al ejecutarlo, pregunte: "¿Es Londres la capital de Inglaterra?" y, a continuación, "Apriete S en caso afirmativo".

Desarrolle el listado para emitir mensajes de acuerdo con la corrección o incorrección de la respuesta.

SOLUCION:

PROGRAMA

```
10 PRINT "¿Es Londres la capital
   de Inglaterra?"
20 PRINT
30 PRINT "Apriete S en caso afir-
   mativo."
40 LET R$ = INKEY$: IF R$ =
   x " " THEN GO TO 40
50 IF R$ = "S" THEN PRINT
   "Correcto": STOP
60 PRINT "Incorrecto"
70 STOP
```

COMENTARIOS

En la línea 40, la ejecución del programa queda en un bucle sobre sí mismo, según se vió en el ejemplo anterior, pero utilizando INKEY$ indirectamente, gracias a R$.
Tanto en la línea 40 como en la 50, R$ sólo puede contener un carácter –el de la tecla apretada, dado por INKEY$– o ninguno –cadena vacía–.

DIDACTICA:

* Recuerde a los alumnos:

- Que en BASIC hay operadores aritméticos, de relación, lógicos y funcionales.
- Cómo y para qué se utilizan los operadores que ya conocen.
- Qué es una cadena vacía.
- Que cada carácter tiene en cualquier computador un valor representado por un número (Código ASCII) que va desde el 32 hasta el 126, ambos inclusive.

* Explique a los alumnos:

- Que los operadores funcionales son algoritmos incorporados al computador. Entre ellos, están CHR$, CODE, LEN, STR$, VAL e INKEY$.
- Que los operadores funcionales operan sobre los datos que entran por teclado, proporcionando un valor como resultado o, también, sobre valores suministrados por el propio programa.
- Que las funciones de estos operadores pueden ser numéricas o alfanuméricas y operativas.
- Que, cuando se utilizan seguidos de una expresión correcta, proporcionan el resultado de la función.
- Que, cuando se utilizan seguidos de una expresión incorrecta, proporcionan un mensaje de error.
- Para qué sirven los operadores funcionales CHR$, CODE, LEN, STR$, VAL e INKEY$.

* Que los alumnos observen experimentalmente:

- Cómo CHR$ (expresión numérica) proporciona el carácter que corresponde a ese valor (número).
- Cómo CODE (expresión alfanumérica) proporciona el valor (número) que corresponde al carácter primero de la expresión alfanumérica. Es decir, que CODE es la función inversa de CHR$.

- Cómo LEN (expresión alfanumérica) proporciona el número de caracteres que tiene la expresión alfanumérica, incluidos los espacios. Es decir, nos da la longitud de la cadena de caracteres.
- Cómo podemos usar LEN en la suma de cadenas o de variables de caracteres.
- Cómo STR$ (expresión numérica) convierte números en cadenas de caracteres, lo cual supone un considerable ahorro de memoria para el computador.
- Cómo VAL sólo puede aplicarse a una expresión alfanumérica que represente a una expresión numérica.
- Cómo VAL (expresión alfanumérica) proporciona el valor numérico de la expresión alfanumérica. Es decir, convierte la expresión alfanumérica en un número.
- Cómo VAL es la función inversa de STR$ en el sentido de que si aplicamos STR$ a una expresión numérica y, luego, aplicamos VAL al resultado obtenido (expresión alfanumérica) volveremos a obtener la expresión numérica primera.

  Pero si aplicamos VAL a una expresión alfanumérica y, luego, aplicamos STR$ al resultado (expresión numérica) no siempre obtendremos la expresión alfanumérica primera.
- Cómo INKEY$ se puede aplicar directamente o a través de una cadena de caracteres. Es decir:

  Si al llegar la lectura del programa a la función INKEY$ no hay ninguna tecla apretada, el computador la considerará como una cadena vacía.
  Pero si hay alguna tecla apretada, el computador proporcionará el carácter de dicha tecla.
- Cómo utilizando INKEY$ se puede impedir que el computador siga la lectura del programa hasta que no se apriete una tecla cualquiera o una tecla determinada, según se haya programado.

EJERCICIOS:

1. Utilizar los operadores funcionales CHR$ y CODE para averiguar:
   1.1. Qué carácter corresponde a cada uno de los valores números) comprendidos entre el Ø y el 255, ambos inclusive (*).
   1.2. Qué valor (número) corresponde a cada uno de los caracteres que hay en el teclado del ordenador en uso (*).

2. Teclear y ejecutar el siguiente programa para ver impresos en la pantalla todos los símbolos, caracteres y comandos de que dispone el ordenador:

```
1Ø FOR A = 32 TO 255
2Ø PRINT CHR$ (A)
3Ø NEXT A
```

3. Averiguar el número de caracteres que componen cada una de estas cadenas:

   "En un lugar de la Mancha "
   "de cuyo nombre no quiero acordarme ... " (*)

4. Escribir el programa que, una vez ejecutado, imprima en pantalla el número de caracteres que contienen en total las dos cadenas anteriores (*).

5. Escribir y ejecutar un programa que imprima en pantalla el resultado de la suma de dos variables de caracteres cualesquiera (*).

6. Aplicar STR$ a las siguientes expresiones numéricas y anotar los resultados: (25), (2*5), (2/5), (2 + 5), (2–5) (*).

7. Aplicar VAL a los resultados obtenidos en el ejercicio anterior y explicar lo que ha ocurrido (*).

8. Teclear el siguiente programa:

```
 5 PRINT "comienza"
1Ø PRINT 5
2Ø PRINT 8
3Ø IF INKEY$ = " " THEN GO TO 3Ø
4Ø PRINT
5Ø PRINT (5 * 8)
6Ø IF INKEY$ = "P" THEN GO TO 5
7Ø STOP
```

Ejecutar el programa anterior y observar lo que pasa:
– Al apretar una tecla cualquiera.
– Al apretar P.

9. Confeccionar un programa que, una vez ejecutado, imprima en pantalla las respuestas a una serie de preguntas sobre un tema determinado; si es o no correcta la respuesta, y que, para poder volver a repetir cada pregunta contestada erróneamente e introducir una nueva respuesta, sea necesario apretar una tecla determinada.
Por ejemplo, decir cuál es la capital de cada uno de los siguientes países: España, Portugal, Francia, Italia, Albania, Yugoslavia, Grecia, Turquía, Siria, Líbano, Israel, Egipto, Libia, Túnez, Argelia y Marruecos (*).

10. Elaborar sus propios programas con los comandos conocidos. Aplicar en los mismos algún operador funcional, por ejemplo, ordenar palabras alfabéticamente, utilizando el comando CODE. Estudie estos ejemplos y saque conclusiones:

    1. Escribir el programa que indique la existencia en kilogramos de las diferentes frutas que hay en un almacén.

    2. Escribir un programa para resolver ecuaciones de primer grado, siendo la incógnita X.

SOLUCIONES

```
  1 REM Ficha:"VAL " Ej.2"
  5 CLS : PRINT "ECUACION DE PRIMER GRADO:"''
 10 INPUT "Ecuacion: ";e$: IF LEN e$=0 THEN  GO TO 10
 20 PRINT e$;" = 0"
 30 LET a=0: LET b=0: LET c=0
 40 FOR x=1 TO 3
 50 IF x=1 THEN  LET a=VAL e$
 60 IF x=2 THEN  LET b=VAL e$
 70 IF x=3 THEN  LET c=VAL e$
 80 NEXT x
 90 IF a=0 OR c=0 OR a=b OR a=c THEN  PRINT "Ecuacion imperf
    ecta": GO TO 10
100 IF NOT INT (a-b)=INT (b-c) THEN  GO TO 130
110 LET x=VAL e$/(c-VAL e$)+x
120 PRINT "x = ";x: GO TO 10
130 PRINT "Es ecuacion de segundo grado": GO TO 10
```

Ejemplo de aplicación de la instrucción de INKEY$:

```
  1 REM Ficha:"INKEY$"
  5 PRINT "Pulse un numero"''
 10 LET A$="0": IF CODE INKEY$<49 OR CODE INKEY$>57 THEN  GO
    TO 10
 20 LET A$=INKEY$
 30 IF CODE A$<49 OR CODE A$>57 THEN  GO TO 10
 40 GO TO VAL A$*100
100 PRINT "Pulsado el 1": GO TO 10
200 PRINT "Pulsado el 2": GO TO 10
300 PRINT "Pulsado el 3": GO TO 10
400 PRINT "Pulsado el 4": GO TO 10
500 PRINT "Pulsado el 5": GO TO 10
600 PRINT "Pulsado el 6": GO TO 10
700 PRINT "Pulsado el 7": GO TO 10
800 PRINT "Pulsado el 8": GO TO 10
900 PRINT "Pulsado el 9": GO TO 10
```

DIM	DIM nombre de la matriz (definición de elementos)	DIMENSIONAR

*** LAS MATRICES EN BASIC ***

INTERPRETACION:

Una matriz en BASIC debe ser concebida como un archivo clasificado de elementos. Estos elementos pueden ser números o caracteres.

Con la sentencia **DIM** se dimensiona **una matriz** denominada "nombre de la matriz" y organizada según las especificaciones dadas en la definición de elementos, reservándose la memoria suficiente para almacenar todos sus elementos. Es decir, **se establecen las dimensiones de esa matriz.**

POSIBILIDADES:

El nombre de la matriz sólo puede ser una letra y la definición de elementos incluye diferentes conceptos, según el tipo de matriz que se pretenda dimensionar. Estos tipos dependerán de la clase de elementos –números o caracteres– y de la organización de los elementos dentro de la matriz.

En una misma matriz sólo puede haber elementos numéricos o, por el contrario, alfanuméricos.

DIM debe ser introducida en el programa antes de pretender utilizar las posibilidades que ofrecen las matrices en BASIC y que estudiamos a continuación.

Para el dimensionado de matrices alfanuméricas –de caracteres–, el nombre de la matriz debe ser una letra seguida del símbolo $. A las matrices numéricas les basta simplemente con la letra.

Los elementos de una matriz pueden organizarse según **una sola dimensión** –secuencia lineal de elementos– o **multidimensionalmente,** distribuyéndolos en filas y columnas e, incluso, más allá como tendremos oportunidad de ver.

Los elementos de cualquier matriz en BASIC son, a todos los efectos, variables cuyo nombre genérico es el mismo para todos y coincide con el nombre de la matriz, distinguiéndose unos de otros por la posición que ocupan dentro de ella.

En el momento de dimensionar una matriz –aplicando la sentencia DIM–, todos los elementos de la misma son Ø. O, dicho de otro modo, las variables que conforman la matriz contienen el valor Ø.

A los elementos de una matriz, también se les conoce por **variables con subíndice**, siendo **el subíndice** el indicador de la posición que ocupan en ella.

Las variables con subíndice pueden manipularse en un programa de la misma forma, y sin ninguna restricción, en que han sido utilizadas las variables en general a lo largo de todo este libro.

Vistos estos conceptos generales, pasemos a unos ejemplos para precisar las ideas.

En este sentido es conveniente aclarar que los ejemplos que se plantean a continuación parten del supuesto de **la necesidad de utilizar matrices** y, consiguientemente, la aplicación de las mismas es inmediata. De aquí podrá el lector sacar conclusiones de tipo operativo que le son imprescindibles, pero –y esto es lo más importante– para usar las matrices en BASIC de una forma eficiente, el programador debe plantearse con claridad el porqué de la matriz que piensa definir.

FORMA DE TECLEAR LA INSTRUCCION

Teclear **DIM, el nombre de la matriz y la definición de los elementos entre paréntesis y separados por comas,** seguido de **ENTER.**

EJEMPLO:

Introducción al planteamiento de matrices numéricas.

Supongamos que es de nuestro interes tener controladas las notas obtenidas por los alumnos de tres centros escolares con cinco cursos de diez alumnos cada centro y, de momento, no es necesario conocer sus nombres, sino sólo sus notas para llegar a conclusiones de tipo estadístico.

En primer lugar, asociaremos uno de los centros al número 1; dentro de ese centro vamos a denominar 1 también al primer curso, 2 al segundo, 3 al tercero, 4 al cuarto y 5 al quinto, y, dentro de cada curso, volvemos a dar al número 1 el significado de primer alumno, al 2 el de segundo alumno, y así, sucesivamente, hasta llegar al décimo con el número 10.

Todo esto lo habremos hecho con la intención de poder saber que, si nos encontramos con una información del tipo (1, 4, 8), estamos refiriéndonos al primer centro, cuarto curso y octavo alumno.

Con el mismo criterio (2, 1, 2) significará que es el segundo alumno del primer curso del segundo centro. Y un (3, 2, 10) vendrá a decirnos que estamos en el tercer centro, segundo curso, décimo alumno.

Gracias a este planteamiento, hemos estructurado los subíndices de nuestra futura matriz de forma que signifique algo congruente para el programador.

Razonamientos de este tipo intuitivo o altamente reflexionado, según la complejidad del asunto, deben ser seguidos para que la organización de los elementos dentro de la matriz responda al criterio de eficacia que se debe exigir.

Bien, una vez en este punto, es relativamente fácil llegar a hacer operativa una matriz.

Comenzaremos por dimensionar la matriz, para la cual debemos darle al ordenador datos que la ponderen. Así, en nuestro ejemplo, con **DIM N** estamos ordenando al computador que dimensione una matriz numérica cuyo nombre sea **N**, pero ahora necesita saber cuántos elementos la van a componer y cómo van a estar distribuidos.

La instrucción completa será **DIM N (3, 5, 1Ø)** –3 centros, 5 cursos, 10 alumnos por curso–, la cual, una vez ejecutada, habrá originado en el computador la creación de **150 variables,** cuyo contenido inicial es **Ø,** de nombre **N** y con subíndices que van desde **(1, 1, 1)** hasta **(3, 5, 10).**

De ahora en adelante, para referirnos a un elemento cualquiera de esta matriz, lo haremos mediante la expresión **N (C, R, A),** de forma que esta notación viene a decir nota del **alumno** A, del **curso R,** del **centro C.**

De forma explícita, las variables de las que disponemos en memoria son:

Correspondientes al primer centro:

N (1, 1, 1) N (1, 1, 2) N (1, 1, 3) N (1, 1, 4) N (1, 1, 5) N (1, 1, 6)
N (1, 1, 7) N (1, 1, 8) N (1, 1, 9) N (1, 1, 10)
N (1, 2, 1) N (1, 2, 2) N (1, 2, 3) N (1, 2, 4) N (1, 2, 5) ..... N (1, 2, 10)
N (1, 3, 1) N (1, 3, 2) N (1, 3, 3) .... N (1, 3, 10)
N (1, 4, 1) N (1, 4, 2) N (1, 4, 3) .... N (1, 4, 10)
N (1, 5, 1) N (1, 5, 2) N (1, 5, 3) .... N (1, 5, 10)

Correspondientes al segundo centro	N (2, 1, 1)	N (2, 1, 2)	N (2, 1, 3)	....	N (2, 1, 10)
	N (2, 2, 1)	N (2, 2, 2)	N (2, 2, 3)	....	N (2, 2, 10)
	N (2, 3, 1)	N (2, 3, 2)	N (2, 3, 3)	....	N (2, 3, 10)
	N (2, 4, 1)	N (2, 4, 2)	N (2, 4, 3)	....	N (2, 4, 10)
	N (2, 5, 1)	N (2, 5, 2)	N (2, 5, 3)	....	N (2, 5, 10)
Correspondientes al tercer centro	N (3, 1, 1)	N (3, 1, 2)	N (3, 1, 3)	....	N (3, 1, 10)
	N (3, 2, 1)	N (3, 2, 2)	N (3, 2, 3)	....	N (3, 2, 10)
	N (3, 3, 1)	N (3, 3, 2)	N (3, 3, 3)	....	N (3, 3, 10)
	N (3, 4, 1)	N (3, 4, 2)	N (3, 4, 3)	....	N (3, 4, 10)
	N (3, 5, 1)	N (3, 5, 2)	N (3, 5, 3)	....	N (3, 5, 10)

Hemos acordado que cualquiera de estas variables está representada por N (C, R, A), lo cual significa que, si hacemos C = 1, R = 2 y A = 4, nos estamos refiriendo a la variable N (1, 2, 4), que figura recuadrada en el esquema anterior y cuyo contenido inicial es Ø.

Ahora ya somos conscientes de lo que significa la primera línea de nuestro programa:

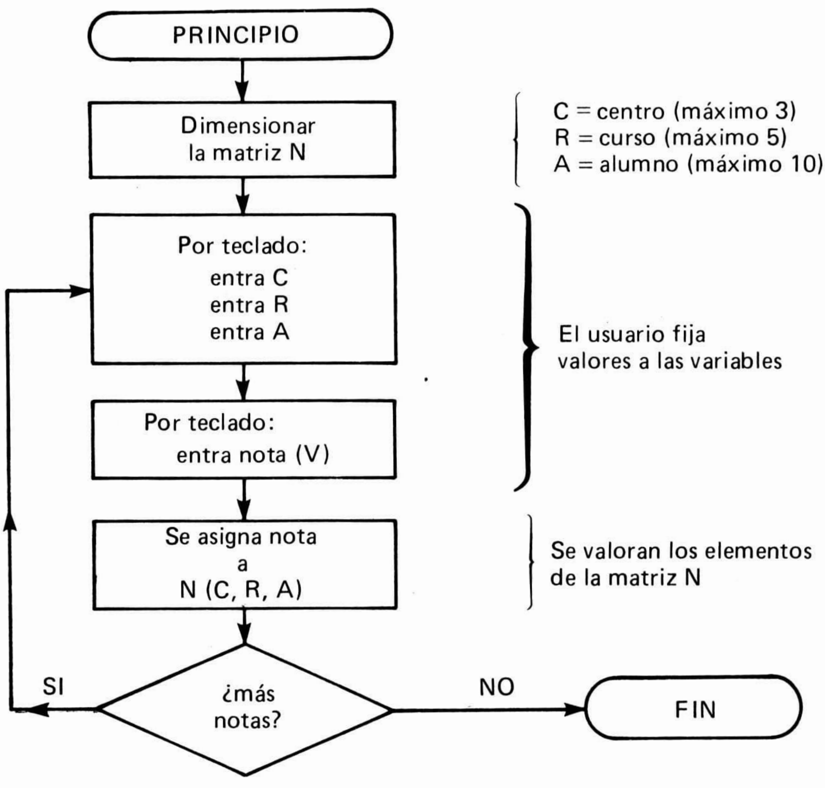
1Ø DIM N (3, 5, 10)

En lo que sigue, haremos operativa esta matriz numérica.

Si quisiéramos darle un valor determinado a un elemento cualquiera de esta matriz, podríamos escribir, por ejemplo,

LET N (1, 2, 4) = 7.5

Pero esta forma de dar valor a las variables de una matriz no suele ser práctica, y, en este caso, no lo es de ningún modo desde el punto de vista del ejemplo propuesto. Resulta más conveniente diseñar un programa que responda a este diagrama:



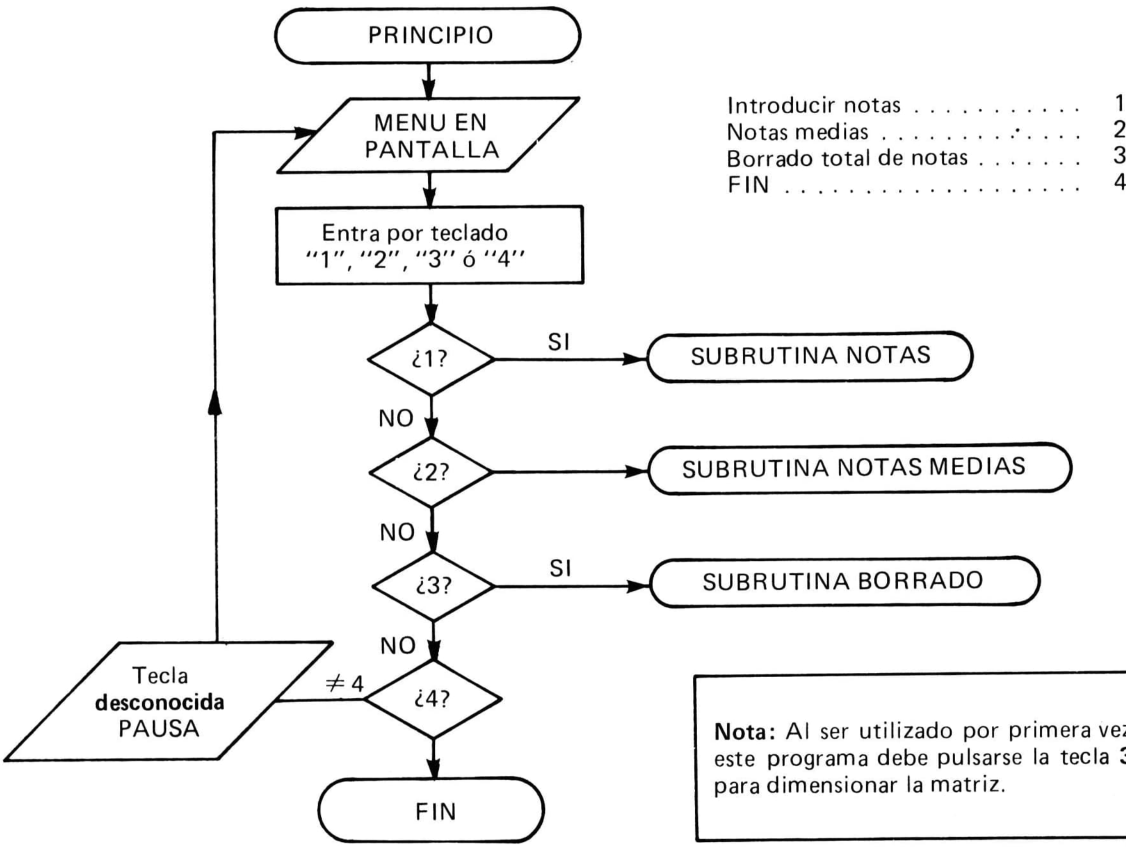
> Un programa con este tipo de diseño no debe ser ejecutado con **RUN/ENTER**, ya que pondría a Ø el contenido de las variables. Es mejor correrlo con **GO TO** a *la línea siguiente a aquélla donde se dimensiona la matriz.*

Un listado que corresponda a la idea anterior puede ser:

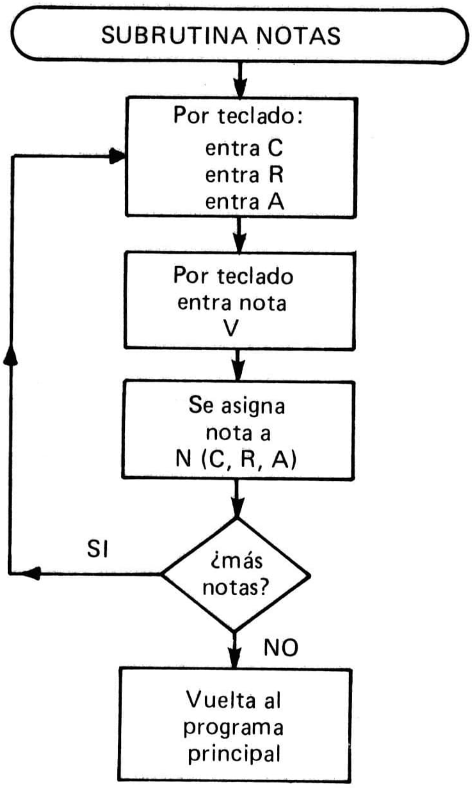
PROGRAMA	COMENTARIOS
1Ø DIM N (3, 5, 1Ø) **2Ø INPUT "Centro 1, 2 ó 3?"; C** **3Ø INPUT "Curso 1, 2, 3, 4 ó 5?";R** **4Ø INPUT "Alumno del 1 al 1Ø?";A** **5Ø INPUT "Nota?"; V** **6Ø LET N (C, R, A) = V** **7Ø INPUT "Más notas? (S/N)"; R$** **8Ø IF R$ = "S" THEN GO TO 2Ø** **9Ø STOP**	Estudie primero y teclee después este programa. Ejecútelo y dé los valores que quiera a **C**, **R**, **A** y **V**. Si suponemos que **C** = **1**, **R** = **3**, **A** = **5** y **V** = **6**, con el comando directo **PRINT N (1, 3, 5)**, obtendremos un **6** en la pantalla. Recuerde que este programa debe ejecutarlo con GO TO 9, línea anterior a la que dimensiona la **matriz.**

Cuando ejecute este programa y haga sus pruebas, seguramente notará que, si bien la matriz se rellena a medida que da valores a **C**, **R**, **A** y **V** —cosa que puede comprobar con el comando directo PRINT seguido de cualquier elemento de la matriz—, no es útil en ningún otro sentido.

Avancemos un poco más con el análisis de los siguientes diagramas:



Nota: Al ser utilizado por primera vez este programa debe pulsarse la tecla **3** para dimensionar la matriz.



COMENTARIOS

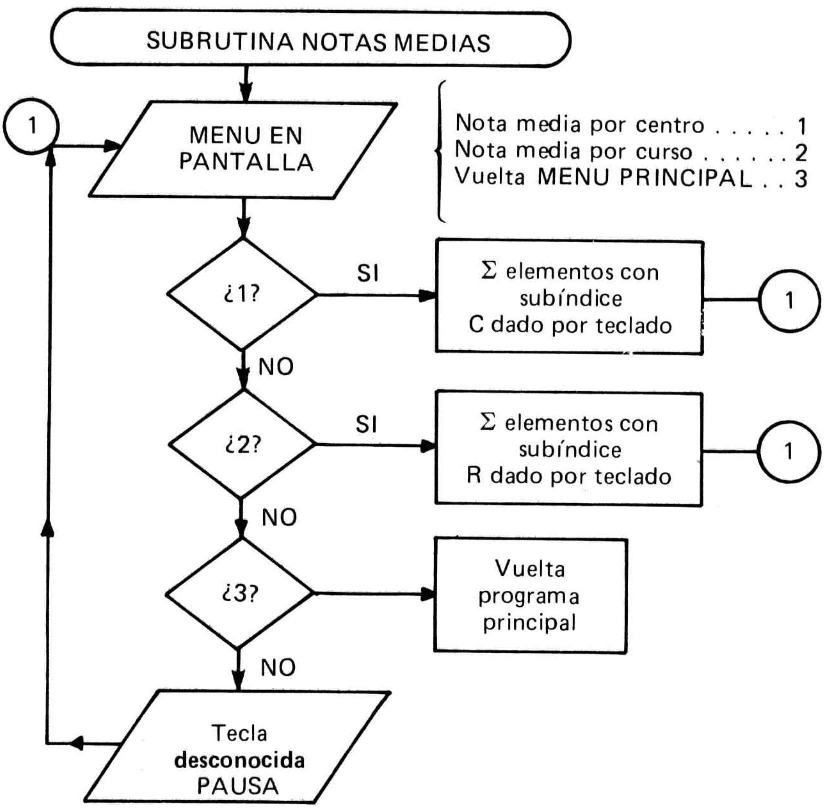
El programa principal, gracias al menú, dirige la secuencia de lectura a una de las tres subrutinas, en función de las necesidades o los deseos del usuario del programa.

Por teclado, entran los valores de las variables C (centro), R (curso) y A (alumno).

También por teclado, entra el valor de la variable V (nota).

Se asigna la nota V al alumno A del curso R del centro C.

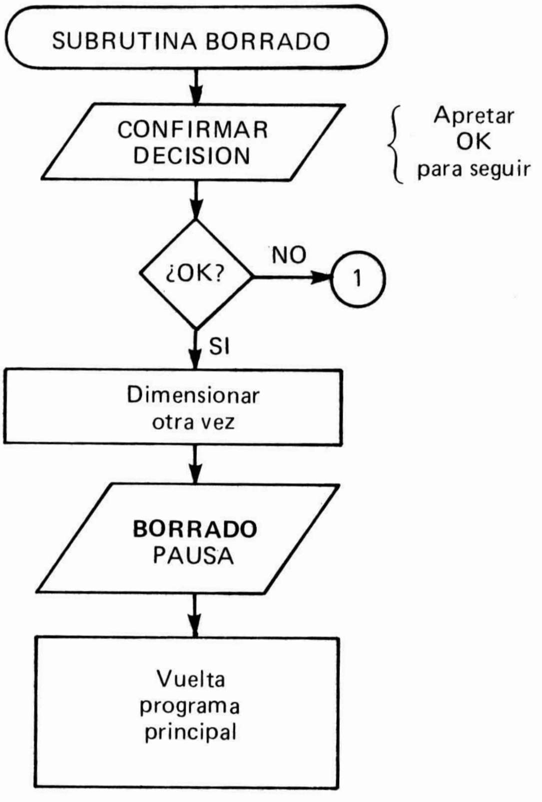
El bucle permite repetir el proceso anterior hasta que se hayan asignado valores a todas las variables o se decida volver al programa principal.



COMENTARIOS

Gracias al menú, el usuario del programa puede obtener la nota media por centro o por curso, o volver al programa principal.

Indica la suma de n elementos con subíndice C o R –introducidos por teclado– dividida por n para obtener la nota media.



COMENTARIOS

Mediante esta subrutina, se confirma la decisión del usuario de borrar los valores de las variables o de volver al programa principal.

Un programa que responda a las características de los diagramas anteriores puede ser el siguiente:

PROGRAMA

COMENTARIOS

Para una mejor presentación del menú y del uso de PRINT en general, vea el apéndice DISEÑO DE PANTALLAS.

```
  1 REM  TO ICHA:" DIM "
 10 PRINT "1 para introducir notas."
 20 PRINT "2 para notas medias."
 30 PRINT "3 para borrado total, o en la primera utilizacion."
 40 PRINT "4 para finalizar el trabajo."
 50 INPUT R$
 60 IF R$="1" THEN  GO SUB 200
 70 IF R$="2" THEN  GO SUB 300
 80 IF R$="3" THEN  GO SUB 700: GO TO 10
 90 IF R$="4" THEN  STOP
100 CLS
110 PRINT "TECLA DESCONOCIDA"
120 FOR X=0 TO 100: NEXT X
130 CLS
140 GO TO 10
190 REM "NOTAS"
```

```
200 CLS
210 INPUT "Centro: ";C
220  INPUT "Curso: ";R
230 INPUT "Alumno?: ";A
240 INPUT "Nota?: ";V
250 LET N(C,R,A)=V
260 INPUT "Mas notas? (S/N): ";R$
270 IF R$="S" OR R$="s" THEN  GO TO 210
280 RETURN
300 REM "NOTAS MEDIAS"
310 CLS
320 PRINT "1 Para nota media por centro."
325 PRINT "2 para nota media por curso."
330 PRINT "3 para volver al menu principal."
340 INPUT R$
350 IF R$="1" THEN  GO TO 420
360 IF R$="2" THEN  GO TO 540
370 IF R$="3"  THEN  RETURN
380 CLS
390 PRINT "TECLA DESCONOCIDA"
400 FOR X=0 TO 100: NEXT X
410 GO TO 310
420 INPUT "Centro a analizar?: ";C
430 LET Z=0
440 FOR R=1 TO 5
450 FOR A=1 TO 10
460 LET S=N(C,R,A)
470 LET Z=Z+S
480 NEXT A
490 NEXT R
500 PRINT ''"La nota media del centro: "'C;" es ";Z/50
510 INPUT "Otra consulta? (S/N): ";R$
520 IF R$="S" OR R$="s" THEN  GO TO 310
530 RETURN
540 INPUT "Curso a analizar?: ";R
550 LET Z=0
560 FOR C=1 TO 3
570 FOR A=1 TO 10
580 LET S=N(C,R,A)
590 LET Z=Z+S
600 NEXT A
610 NEXT C
620 PRINT ''"La nota media de los cursos: "'R;" es ";Z/30
630 GO TO 510
700 REM "BORRADO Y PRIMERA VEZ"
710 CLS
720 PRINT "  Esta opcion del menu solo debeusarla cuando se
    utiliza "
725 PRINT "el pro-grama  por  primera vez, o  paraborrar tod
    as las notas."
```

```
730 INPUT "Si lo confirma teclee O.K.: ";R$
740 IF R$<>"O.K." THEN  CLS : GO TO 10
750 DIM n(3,5,10)
760 PRINT "Matriz inicializada"
770 FOR X=0 TO 10: NEXT X
780 CLS : RETURN
```

¡OJO! Si su ordenador no dispone del comando CLS, no utilice CLEAR, ya que si lo usa pondrá todas las variables a Ø. Busque o diseñe una subrutina para sustituirlo.

En la línea 12Ø se ha generado un bucle de retardo que durará más cuanto mayor sea el valor final de X.

Si su ordenador permite el uso de un comando de retardo, por ejemplo, PAUSE, aplíquelo.

EJEMPLO:

Introducción al planteamiento de matrices alfanuméricas.

En esta ocasión vamos a suponer que nos interesa conocer los nombres de los alumnos de los tres centros anteriores.

Antes de entrar propiamente en la resolución de este ejemplo, es conveniente conocer las peculiaridades de las matrices de caracteres de forma paulatina.

De momento, vamos a centrarnos en colocar dentro de una matriz de caracteres los nombres de los alumnos de un curso de un solo centro. Supongamos que sus nombres son: Juan, Luis, Francisco, Angel, Miguel, Tomás, Federico, Félix, José y Andrés.

Como vemos, son diez cadenas de caracteres que tienen:

4 caracteres la primera: "**Juan**"
4 caracteres la segunda: "**Luis**"
9 caracteres la tercera: "**Francisco**"
5 caracteres la cuarta: "**Angel**"
6 caracteres la quinta: "**Miguel**"
5 caracteres la sexta: "**Tomás**"
8 caracteres la séptima: "**Federico**"
5 caracteres la octava: "**Félix**"
4 caracteres la novena: "**José**"
6 caracteres la décima: "**Andrés**"

Comparando la longitud de todas ellas, llegamos a la conclusión evidente que la cadena más larga es la tercera, "**Francisco**", que tiene **9 caracteres.**

Así, pues, tenemos **10 cadenas** a guardar en una matriz y la longitud de la mayor es de **9 caracteres**. Estos son dos datos necesarios para dimensionar esa matriz. Es decir, si a lo largo del listado de un programa nos encontramos, por ejemplo, con:

... DIM N$ (1Ø , 9) (1)

debemos interpretar que se dimensiona una matriz alfanumérica, denominada **N$**, capaz de contener **10 cadenas** de hasta **9 caracteres** cada una.

Cuando se dimensiona una matriz alfanumérica, se reserva en memoria un espacio para tantos caracteres como resulten de multiplicar el número de cadenas por el número de caracteres que –como máximo– vayan a conformarlas, e, inicialmente, su contenido son **espacios en blanco.**

(1) Hay dialectos del BASIC que no necesitan el segundo parámetro.

Para ver esto con claridad, consideremos **A** cadenas de **L** letras o caracteres en general y dimensionemos la matriz adecuada:

```
70 ..................
80 ... DIM N$ (A, L)
90 ..................
```

En el momento de ejecutarse esta sentencia, se guardan **A * L** espacios. Esto quiere decir que, si tecleamos, por ejemplo, **PRINT N$ (n)** –siendo **n** mayor o igual que **1** y menor o igual que **A**– en pantalla no aparecería nada, que es justamente lo que se espera de un espacio. Pero si el comando fuera **PRINT CODE N$ (n),** obtendríamos **32**, que es el código ASCII del **espacio.**

Con esto, venimos a comprobar que, mientras las matrices numéricas contienen **ceros** inicialmente, las matrices alfanuméricas contienen inicialmente **espacios.**

Si en una matriz se dimensiona **DIM N$ (A, L),** se introduce una cadena de longitud menor que **L**, entonces, los caracteres sobrantes se mantienen en espacios.

Si **L** es menor que la longitud de la cadena, se ignoran los caracteres sobrantes.

Volvamos al caso del curso de diez alumnos cuyos nombres conocemos y cabe preguntarse: *¿Cómo se rellena la matriz alfanumérica una vez diseñada?*

Podemos seguir un proceso similar al de las matrices numéricas. Veamos el siguiente listado:

PROGRAMA	COMENTARIOS
1Ø DIM N$ (1Ø, 9)	Dimensionamos la matriz N$ de 10 cadenas de hasta 9 caracteres.
2Ø INPUT "Número del alumno? "; A	Se pide número de referencia del alumno que corresponderá a su posición en la matriz. Por tanto, debe ser **1 = < A = < 10.**
3Ø INPUT "Nombre? "; L$	Los caracteres del nombre deben ser **1 = < L = < 9.**
4Ø LET N$ (A) = L$	A tendrá el valor numérico dado en 2Ø, y L$ será la cadena definida en 3Ø.
5Ø INPUT "Otro nombre? (S/N) "; R$	Con las líneas 5Ø, 6Ø y 7Ø establecemos las condiciones para repetir el bucle si hay más nombres para introducir en la matriz.
6Ø IF R$ = "S" THEN GO TO 2Ø	
7Ø STOP	

Al ejecutar este programa y dar a **A** los valores **1, 2, 3, ... 10** y a **L$ los nombres** de los diez alumnos citados (Juan, Luis, etc.), se conseguirá rellenar la matriz **N$** de acuerdo con nuestros deseos. Para comprobarlo, basta ejecutar un comando directo del tipo **PRINT N$ (2)/ENTER** y obtendremos en pantalla **Luis.** Haga otras pruebas.

Nuestro siguiente paso será controlar los cinco cursos de un centro. Para ello, bastará con dimensionar la matriz de la línea 10 de la siguiente forma:

```
..................
... DIM N$ (5, 1Ø, 9)
..................
```

en el supuesto de que no haya ningún alumno cuyo nombre tenga más de **9** caracteres en

ninguno de los 5 cursos. Con que sólo existiera uno cuyo nombre tuviese, por ejemplo, 12 caracteres, tendríamos que modificar la anterior matriz de la siguiente forma:

. . . DIM N$ (5, 1Ø, 12)

claro está, a no ser que no tuviera ninguna importancia perder los tres últimos caracteres en cuestión.

Con la instrucción de la línea 10, el ordenador ha reservado en la memoria espacio para 5 * 10 * 9 caracteres y, todos ellos, están inicialmente ocupados por **espacios.**

Un programa adaptado a la nueva situación puede ser:

PROGRAMA	COMENTARIOS
1Ø DIM N$ (5, 1Ø, 9)	Dimensiona 5 grupos de 10 cadenas de 9 caracteres cada una.
2Ø INPUT "Curso? "; R	1 = < R = < 5
3Ø INPUT "Número del alumno? "; A	**1 = < A = < 10**
4Ø INPUT "Nombre? "; L$	**1 = < LEN L$ = < 9**
5Ø LET N$ (R, A) = L$	R tendrá el valor numérico dado en 2Ø, y A el dado en 3Ø; L$ está definida en 4Ø.
6Ø INPUT "Otro nombre? (S/N) "; R$	
7Ø IF R$ = "S" THEN GO TO 2Ø	
8Ø STOP	

Ahora estudiamos el caso completo de los tres centros con cinco cursos cada uno de diez alumnos.

La forma de dimensionar la matriz parece evidente en función de lo expuesto hasta aquí. Se trata de tres grupos formados por cinco subgrupos cada uno que tienen diez cadenas de nueve caracteres cada una. Todo ello corresponde a la siguiente instrucción:

. . . DIM N$ (3, 5, 1Ø, 9)

que nos llevaría al siguiente programa:

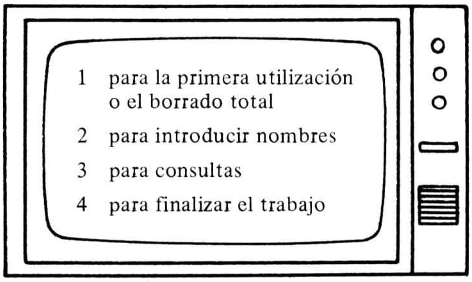
PROGRAMA	COMENTARIOS
1Ø DIM N$ (3, 5, 1Ø, 9)	Dimensionado de la matriz N$ según lo comentado.
2Ø INPUT "Centro? "; C	**1 = < C = < 3**
3Ø INPUT "Curso? "; R	**1 = < R = < 5**
4Ø INPUT "Número del alumno? "; A	1 = < A = < 10
5Ø INPUT "Nombre? "; L$	**1 = < LEN L$ = < 9**
6Ø LET N$ (C, R, A) = L$	C tendrá el valor numérico dado en 2Ø; R el dado en 3Ø, y A el dado en 4Ø; L$ está definida en 5Ø.
7Ø INPUT "Otro nombre? (S/N) "; R$	
8Ø IF R$ = "S" THEN GO TO 2Ø	
9Ø STOP	

NOTA: Recordamos al lector que el tratamiento de matrices varía de un BASIC a otro. Aquí hemos planteado el caso del más completo, de forma tal que cualquier otro le sea de fácil asimilación con respecto a éste.

Una vez tecleado y ejecutado este programa, damos valores a todas las variables de la matriz hasta rellenar las 3 * 5 * 10 cadenas.

Una vez hecho esto, supongamos que queremos saber el nombre del alumno cuya referencia (matrícula, por ejemplo) es 10, del curso 4 del centro 2. Para ello, podríamos emplear el comando directo **PRINT N$ (2, 4, 10)** y obtendríamos en pantalla el nombre deseado.

Esta forma de averiguar el contenido de los diferentes elementos de la matriz no es muy práctica. Por esta razón, sugerimos al lector el ejercicio de completar el anterior listado de forma parecida a lo establecido en los diagramas de las matrices numéricas, basados en **menús** presentados en pantalla, de los cuales damos una sugerencia en el siguiente esquema:



El lector debe observar que un listado que contenga una combinación adecuada de las instrucciones dadas en el último ejemplo de las matrices numéricas y del que se deriva el comentario anterior, le puede llevar a obtener resultados en los que se unan el nombre del alumno y su nota, utilizando

PRINT N (C, R, A) y **PRINT N$ (C, R, A)**

Este tema, por sí mismo, abriría la posibilidad de una nueva publicación delicada a los ficheros en BASIC, que se sale de los límites que nos hemos propuesto en la presente.

DIDACTICA:

* Recordar a los alumnos:
  - Los conceptos de constante y variable.
  - Qué es un bucle y cómo se introduce en un programa mediante las sentencias FOR ... TO/NEXT y FOR ... TO ... STEP/NEXT.
  - La utilidad de los comandos estudiados, especialmente INPUT, LET, GO TO, GO SUB/RETURN e IF/THEN.
* Explicar a los alumnos:
  - Qué es una matriz y para qué sirve.
  - Qué son matrices numéricas y matrices alfanuméricas, y cómo se nombran.
  - Cómo se plantean las matrices numéricas y las alfanuméricas.
  - Qué es dimensionar una matriz y qué son matrices unidimensionales y matrices multidimensionales.
  - Qué son las variables con subíndice y qué indican los subíndices.
  - Para qué y cómo se utiliza la sentencia DIM.
* Que los alumnos observen experimentalmente:
  - Cómo se dimensiona una matriz numérica.

- Cuántas variables numéricas tiene una matriz numérica dada.
- Cómo se dimensiona una matriz alfanumérica.
- Cuántos caracteres contiene una matriz alfanumérica dada.
- Cómo se pone a Ø el contenido de todas las variables de un programa.
- Cómo se corre un programa utilizando GO TO.

EJERCICIOS:

1. Dimensionar las siguientes matrices numéricas:
   1.1. La que permita almacenar 12 variables numéricas.
   1.2. La matriz bidimensional que permita almacenar 12 variables numéricas.
   1.3. La matriz multidimensional (tres dimensiones) que permita almacenar 12 variables numéricas.

2. Escribir el programa que permita valorar las 10 variables numéricas de una matriz bidimensional.

3. Adaptar el listado del programa anterior de forma que, sin modificar el valor de las variables y automáticamente, se valoren las variables de la matriz DIM W (2,5) con la condición de que los sucesivos valores de la matriz DIM A (2,6) se multipliquen por 2 si son mayores que 50, y se eleven al cuadrado si son menores, transfiriéndose con esta condición a la matriz DIM W (2,6).

4. Escribir y ejecutar el siguiente programa cuyo objetivo es determinar el valor total de la venta de diferentes unidades de 3 artículos distintos que se venden, respectivamente, a los precios unitarios de 10, 20 y 30 pesetas.

Hacer los comentarios oportunos al programa.

PROGRAMA

```
 1  LET C = Ø
 5  DIM P (3)
1Ø  DIM T (3)
15  FOR X = 1 TO 3
2Ø  PRINT "Precio del artículo "; X
25  INPUT P (X)
3Ø  NEXT X
35  CLS
4Ø  PRINT "Artículo número ? "
5Ø  INPUT N
55  PRINT "Unidades vendidas? "
6Ø  INPUT V
65  IF P (N) > Ø THEN LET T = P (N) * V
70  LET C = C + 1
75  LET T (C) = T
8Ø  PRINT "Otro artículo ? "
85  INPUT R$
9Ø  IF R$ = "S" THEN GO TO 35
95  PRINT AT 21,Ø; "Total venta  ", T (1) + T (2) + T (3)
```

5. Dimensionar las siguientes matrices alfanuméricas:
   5.1. La que permita almacenar 5 variables de 6 caracteres.
   5.2. La que permita almacenar 6 variables de 5 caracteres.

6. Interpretar el siguiente programa:

PROGRAMA

```
10  DIM A$ (4,7)
20  DIM B$ (4,7)
30  FOR X = 1 TO 4
40  LET B$ (5-X) = A$ (X)
50  NEXT X
```

7. Escribir un programa que ordene alfabéticamente 4 palabras que no tengan más de 7 caracteres cada una.

8. Seguir las instrucciones que se dan para escribir un programa que permita conocer los nombres del tutor y el delegado de cada uno de los seis cursos de un centro escolar.

INSTRUCCIONES

30 Dimensionar una matriz alfanumérica de 6 variables de hasta 14 caracteres cada una cuyo prefijo es F$.
40 Dimensionar una matriz alfanumérica de 6 variables de hasta 15 caracteres cada una cuyo prefijo es J$.
50 Ordenar la impresión de "Curso? "
60 Esperar la entrada de un número correspondiente a un curso.
70 Ordenar la impresión del número anterior a continuación de "Curso? ". Recuerde, el uso del punto y coma.
80 Ordenar imprimir "Tutor? ".
90 Esperar la entrada del nombre del tutor.
100 Ordenar imprimir el nombre del tutor a continuación de "Tutor? ".
110 Ordenar imprimir "Delegado? ".
120 Esperar la entrada del nombre del delegado.
130 Ordenar imprimir el nombre del delegado a continuación de "Delegado? ".
140 Ordenar que el programa vuelva a la línea 50.

9. Elaborar sus propios programas con matrices numéricas y alfanuméricas.

SOLUCIONES

1.1.
```
 1 REM Ficha:" DIM " Ej.1.1"
10 DIM A(12)
```

1.2.
```
 1 REM Ficha:" DIM " Ej.1.2"
10 DIM A(2,6)
```

1.3.
```
 1 REM Ficha:" DIM " Ej.1.3"
10 DIM A(2,3,2)
```

2.
```
 1 REM Ficha:" DIM " Ej.2"
10 DIM A(2,6)
20 FOR X=1 TO 2
30 FOR Y=1 TO 6
40 PRINT "Valor ";X; ",";Y;" de la matriz A=";
50 INPUT A(X,Y): PRINT A(X,Y)
60 NEXT Y
70 NEXT X
```

3.
```
 1 REM Ficha:" DIM " Ej.3"
 5 DIM W(2,6)
10 DIM A(2,6)
20 FOR X=1 TO 2
30 FOR Y=1 TO 6
```

```
 40 IF A(X,Y)<=50 THEN GO TO 60
 50 LET W(X,Y)=A(X,Y)*2
 60 LET W(X,Y)=A(X,Y)^2
 70 NEXT Y
 80 NEXT X
```

5.1.

```
  1 REM Ficha:" DIM " Ej.5.1"
 10 DIM a$(5,6)
```

5.2.

```
  1 REM Ficha:" DIM " Ej.5.2"
 10 DIM A$(6,5)
```

7.

```
  1 REM Ficha:" DIM " Ej.7"
 10 DIM A$(4,7)
 20 FOR X=1 TO 4
 30 INPUT A$(X)
 40 NEXT X
 50 IF A$(1)<=A$(2) THEN GO TO 90
 60 LET B$=A$(1)
 70 LET A$(1)=A$(2)
 80 LET A$(2)=B$
 90 IF A$(2)<=A$(3) THEN GO TO 140
100 LET B$=A$(2)
110 LET A$(2)=A$(3)
120 LET A$(3)=B$
130 GO TO 50
140 IF A$(3)<=A$(4) THEN GO TO 190
150 LET B$=A$(3)
160 LET A$(3)=A$(4)
170 LET A$(4)=B$
180 GO TO 90
190>PRINT A$(1)
200 PRINT A$(2)
210 PRINT A$(3)
220 PRINT A$(4)
```

8.

```
  1 REM Ficha:" DIM " Ej.8
 30 DIM F$(6,14)
 40 DIM J$(6,15)
 50 PRINT "Curso? ";
 60 INPUT C
 70 PRINT C
 80 PRINT "Tutor? ";
 90 INPUT F$(C)
100 PRINT F$(C)
110 PRINT "Delegado? ";
120 INPUT J$(C)
130 PRINT J$(C)
140 GO TO 50
```

LIST	LIST número de línea	LISTAR EN PANTALLA
LLIST	LLIST número de línea	LISTAR EN IMPRESORA

INTERPRETACION:

LIST ordena listar en pantalla todas las líneas del programa a partir del número de línea.

Con LLIST se consigue el mismo resultado pero impreso en papel, mediante una impresora conectada al computador.

POSIBILIDADES:

Si el número de línea no se da, se obtiene un listado del programa desde el principio hasta el final.

En algunos tipos de BASIC, se ofrece la opción de indicar al computador no sólo el número de línea a partir del cual se desea el listado, sino también el último, de forma que se consiguen listados parciales automáticamente.

FORMA DE TECLEAR LAS INSTRUCCIONES

Teclear **LIST o LLIST, seguido de ENTER**, para obtener *el listado completo del programa* en pantalla o en papel.

Teclear **LIST o LLIST y el número de línea**, seguido de *ENTER, para obtener el listado del programa desde la línea del número dado,* en pantalla o en papel.

LOAD	LOAD "nombre"	CARGAR

INTERPRETACION:

Instruye al computador para que cargue en su memoria RAM la información procedente de un soporte de memoria externo. Dicha información ha sido guardada o salvada previamente en ese soporte (cinta magnética o disco) con el "nombre".

POSIBILIDADES:

Normalmente, todos los dialectos del BASIC permiten una modalidad de carga de programas y, una vez terminada la misma, la ejecución automática.

Para que tal posibilidad sea operativa, se requiere cierto método en el proceso de salvar el programa (ver SAVE) o, simplemente, añadir un comando a LOAD.

Cuando se ejecuta esta instrucción, todas la variables son borradas y el programa existente en la memoria RAM, si lo hubiere, desaparece.

FORMA DE TECLEAR LA INSTRUCCION

Teclear **LOAD y comillas dos veces** (" ") o LOAD y el nombre del programa entre comillas, **seguido de ENTER.**

SAVE	SAVE "nombre"	SALVAR

INTERPRETACION:

Transfiere la información contenida en la memoria RAM de un computador a un soporte de memoria externo, **salvándola** o **guardándola** para ser utilizada posteriormente, transfiriéndola nuevamente al computador mediante LOAD.

Es necesario recordar que la memoria RAM no es permanente y, al desconectar el computador de la corriente, toda la información que contiene se pierde. Gracias a SAVE y a la función que cumple, la información queda salvada, bien sobre cinta magnética, bien sobre disco.

POSIBILIDADES:

El MANUAL del ordenador que se esté utilizando indicará los pasos a seguir durante la operación SAVE para provocar la autoejecución de un programa, tras su carga en la memoria, en el caso de requerirla.

Ciertas modalidades de BASIC permiten proteger la información salvada en disco añadiendo un comando a SAVE.

FORMA DE TECLEAR LA INSTRUCCION

Teclear **SAVE** o **SAVE y el nombre del programa entre comillas**. seguido de **ENTER.**

NEW — NUEVO

INTERPRETACION:

Borra el programa que esté en la memoria RAM del computador y todas las variables.

POSIBILIDADES:

Cuando se ejecuta la instrucción LOAD, el resultado –con respecto a un programa que puediera haber en la memoria RAM– es el mismo que si se ejecuta previamente un NEW.

FORMA DE TECLEAR LA INSTRUCCION

Teclear **NEW**, seguido de **ENTER**.

DIDACTICA:

* Recuerde a los alumnos que en BASIC existen sentencias auxiliares que permiten realizar ciertas operaciones o procesos.

* Explique a los alumnos:

  – Que cada computador dispone de diferentes sentencias auxiliares, por lo que deben consultar el MANUAL del mismo para conocerlas y poder aplicarlas.
  – Que las sentencias auxiliares LIST, LLIST, LOAD, SAVE y NEW son comunes a todos los computadores.
  – Para qué sirven estas sentencias auxiliares y cómo se utilizan.

* Que los alumnos observen experimentalmente cómo se aplican las sentencias auxiliares LIST, LLIST, LOAD, SAVE y NEW y aquéllas otras que pueda tener el computador en uso.

EJERCICIOS:

Cuantos el profesor estime oportunos para que los alumnos consigan dominar las sentencias auxiliares de que dispone el computador utilizado.

Apéndices

I. ESQUEMA DE COMANDOS BASIC

II. DIAGRAMAS DE FLUJO

III. ORGANIZACION DE PANTALLAS

IV. FUNCIONES LOGICAS APLICADAS

AND.	Y lógico
OR.	O lógico
NOT.	NO lógico

V. TROCEADO DE CADENAS

VI. ASCII

VII. SISTEMAS DE NUMERACION

- SISTEMA DECIMAL
- SISTEMA BINARIO
- SISTEMA HEXADECIMAL

Apéndice I — ESQUEMA DE COMANDOS BASIC

Operadores aritméticos: ↑, .*, /, +, –, ( )

Operadores de relación: =, <,>, < >, <=, >=

Operadores lógicos: NOT, AND, OR

Operadores funcionales:

- *Operativos:* *ATTR, IN, PEEK*
- *Alfanuméricos:* CHR$, CODE o ASC, INKEY$, LEN, STR$, VAL, VAL$, ABS, ACS, ASN, ATN
- *Numéricos:* COS, EXP, INT, LN ó LOG, RND, SGN, SIN, SQR, TAN

Sentencias:

- *Auxiliares:* *BEEP, BORDER, BRIGHT, CIRCLE,* CLEAR, CLS, CONTINUE, *COPY, DRAW, FLASH, INK, INVERSE,* LIST, LLIST, LOAD, LPRINT, *MERGE,* NEW, *OVER, PAPER,* PAUSE, *PLOT,* PRINT, REM, RUN, SAVE, STOP, *VERIFY*
- *De asignación:* DATA, DEF FN, DIM, INPUT, LET, RANDOMIZE, READ, RESTORE
- *De bifurcación incondicional:* GO SUB, GO TO, RETURN
- *De selección:* IF/THEM
- *De repetición:* FOR ... TO/NEXT, FOR ... TO ... STEP/NEXT

NOTA: Los comandos en cursiva no se explican en esta publicación, ya que sus funciones se escapan de los límites de la misma.

Apéndice II — DIAGRAMAS DE FLUJO

Programar es una actividad lógica que requiere claridad y precisión en todas sus etapas. De entre ellas, la más importante es la que se refiere al planteamiento del propio programa.

Definir los objetivos a lograr y hacerlo de forma certera ahorra esfuerzos inútiles. Para ello, debe plantearse la siguiente pregunta:

¿Que se pretende conseguir con el programa?

La respuesta a esta cuestión no siempre es fácil y, a mayor complejidad, necesitará más tiempo de reflexión.

Una vez superada esta fase, aparece indefectiblemente otro interrogante:

¿Cómo organizar el programa?

Es claro que una forma de organizar un programa consiste en sentarse ante el computador y comenzar a teclear de manera que, finalmente, el listado *obligue* a la máquina a dar una respuesta, aunque no se haya dado una estructura lógica al trabajo.

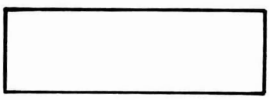
En este APENDICE, vamos a tratar **los diagramas de flujo** u **organigramas**, gracias a los cuales se obtiene un boceto, a grandes rasgos, de lo que será el programa.

La mayor ventaja que se deriva de la realización de *los diagramas de flujo* es el esfuerzo de síntesis al que se somete el programador, con la consiguiente claridad de ideas que de esto se deriva.

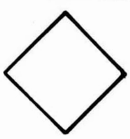
Un diagrama de flujo consiste, en definitiva, en una serie de **símbolos estándar**, unidos por **flechas** para indicar los caminos que la secuencia del programa puede tomar.

Cada *símbolo estándar* implica, por su forma, una determinada actividad, pudiéndose escribir dentro de ellos todo lo que sirva para darlos mayor claridad.

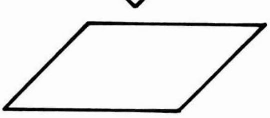
Los símbolos estándar generalmente utilizados en *los diagramas de flujo* son:



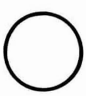
Con **el rectángulo** se expresa la idea de acción, que será llevada a cabo por un comando BASIC.



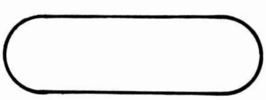
Con **el rombo** se exige la toma de decisión entre dos posibilidades.



Con **este paralelogramo** se indica que entran datos en el computador o salen de él.



Con **el círculo** se conectan las partes del programa que no lo están por medio de flechas, para lo que basta escribir en su interior la referencia oportuna.



Con **esta figura compuesta** se representan el principio y el final de determinadas secuencias, ya sean programas completos o subrutinas.

Antes de pasar a unos ejemplos que clarifiquen lo expuesto hasta aquí, quisiéramos hacer hincapié en que **los organigramas no son consecuencia de los programas, sino que la estructura de cada programa es una consecuencia del organigrama correspondiente.**

Los *diagramas* que figuran a continuación deberán ser hechos a mano por el programador a fin de sacar al lector de cualquier rigor academicista.

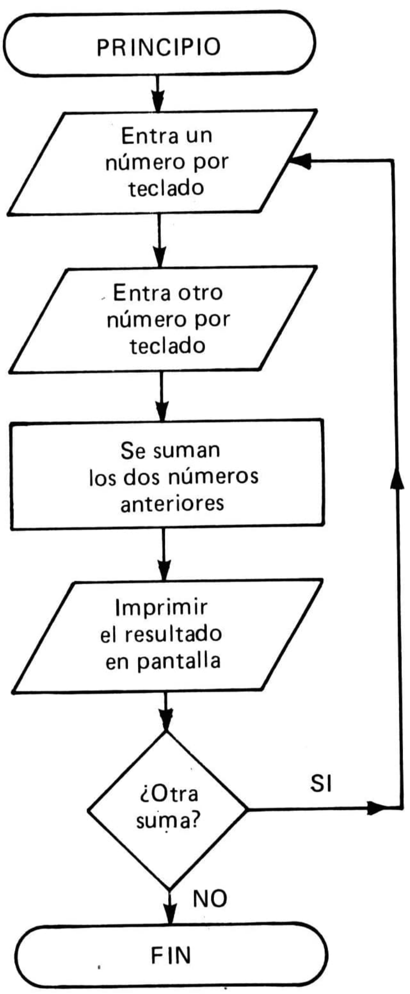
Los diagramas y sus símbolos están pensados para ser usados libremente por el programador de tal forma que se conviertan en un buen útil de trabajo que pueda manejarse sin ninguna rectricción.

Dicho esto, pasemos a analizar algunos problemas elementales susceptibles de ser convertidos en programas.

EJEMPLO:

1. Desarrollar un programa que permita sumar dos números introducidos por teclado, en una secuencia ininterrumpida, y se impriman los resultados en pantalla.

DIAGRAMA:



PROGRAMA:

```
5 REM Sumar dos números cualesquiera

1Ø INPUT A

2Ø INPUT B

3Ø LET C = A + B

4Ø PRINT C

5Ø INPUT "Otra suma?"; S$

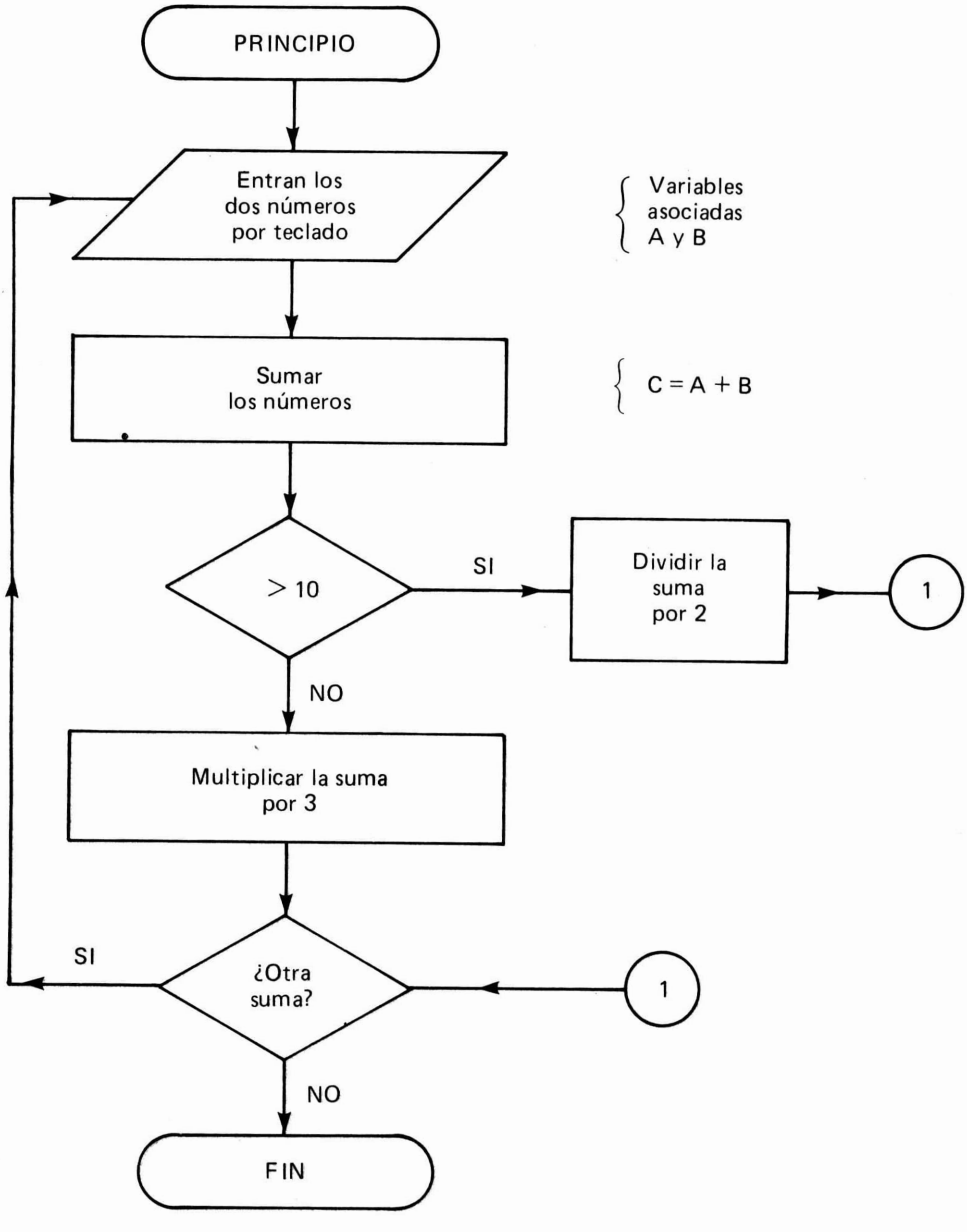
6Ø IF S$ = "S" THEN GO TO 1Ø

7Ø STOP
```

EJEMPLO:

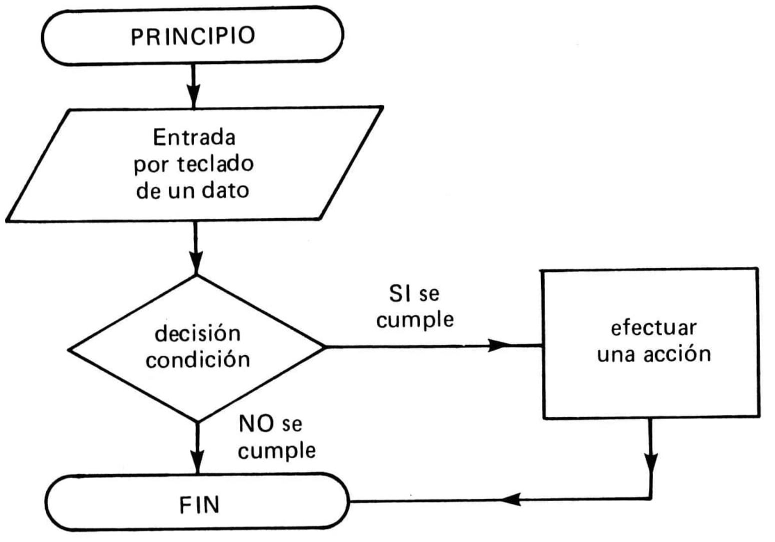
2. Desarrollar un programa que, al ejecutarlo, permita introducir dos números por teclado y si su suma es mayor que diez, esta se divida por dos; en otro caso, que se multiplique por 3.

DIAGRAMA:



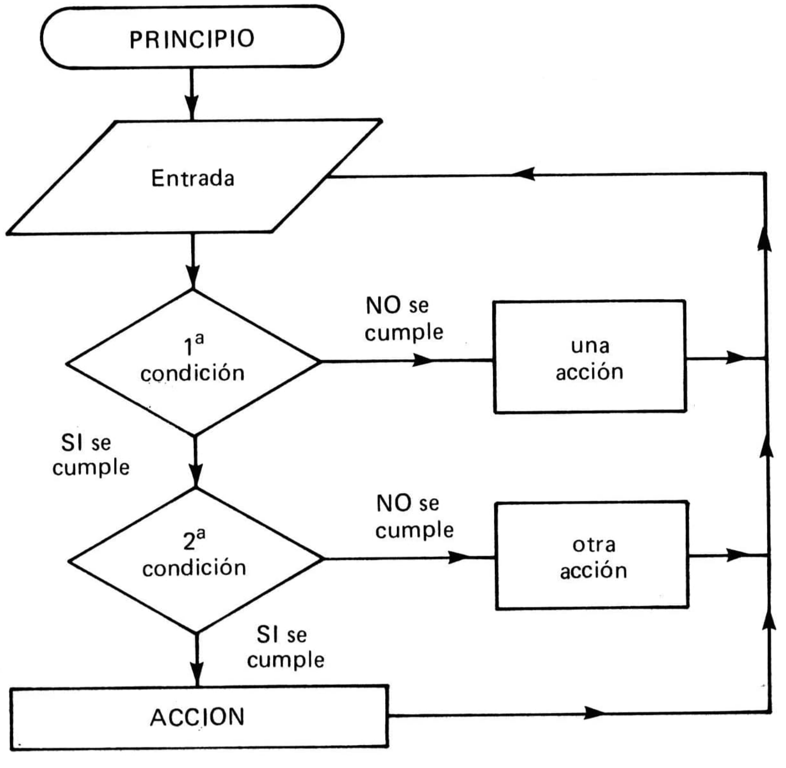
EJEMPLO:

3. Diagrama para una decisión sencilla:



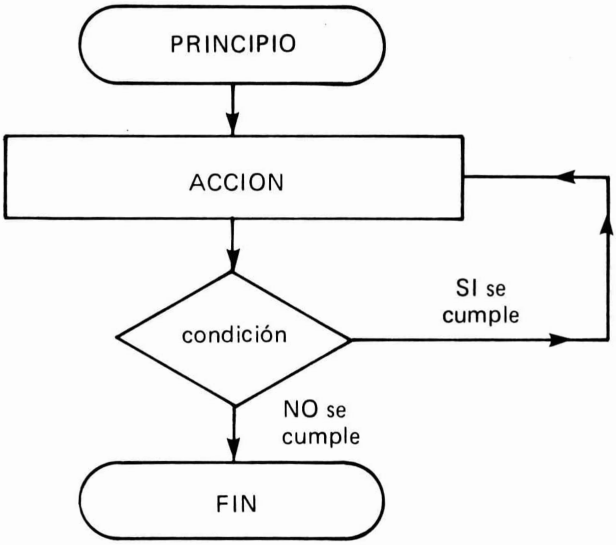
EJEMPLO:

4. Diagrama para más de una decisión y bucle:



EJEMPLO:

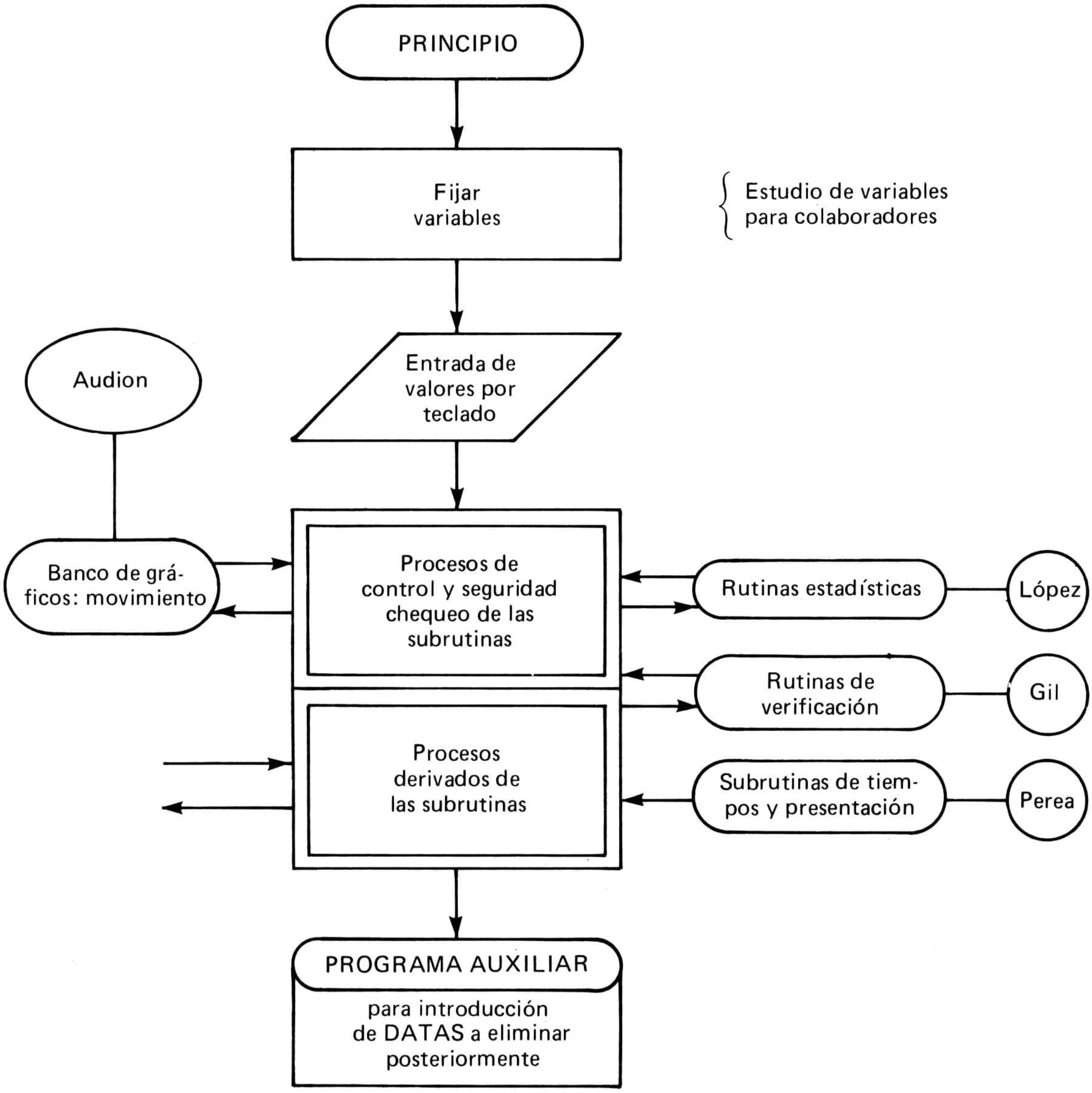
5. Diagrama para un bucle indefinido hasta que no se cumpla una condición:



A medida que los conocimientos de BASIC del programador aumentan, su capacidad para resolver problemas se hace paulatinamente mayor y, en consecuencia, los diagramas tendrán que plantearse desde perspectivas que abarquen mayores generalidades del problema a resolver, del cual se sacarán, a su vez, unos diagramas más precisos.

EJEMPLO:

6. Posible diagrama para un programa de pruebas objetivas:



Todos estos bloques son en sí mismos un programa diferente cada uno, por lo que algunos de ellos los van a realizar los programadores cuyos nombres figuran anejos a los mismos, con el fin de hacer más fácil la tarea.

Cada uno de estos colaboradores debe presentar su propio diagrama al coordinador del proyecto para poder estudiar conjuntamente los problemas que se presentan y las incompatibilidades que puedan surgir. Esta suele ser la forma de trabajar en programas complejos.

DIDACTICA:

* Recordar a los alumnos:

  – Que los diagramas de flujo son representaciones gráficas de los programas mediante símbolos convencionales.
  – Que los diagramas de flujo sirven para organizar la estructura de los programas y para saber los pasos que sigue el ordenador en su ejecución.
  – Que la estructura de un programa es consecuencia del diagrama de flujo correspondiente y no el diagrama la consecuencia del programa.
  – Que los diagramas de flujo sirven para adquirir hábitos que permiten seguir procesos lógicos en la solución de los problemas.

* Explicar a los alumnos:

  – Qué símbolos estándar se utilizan en los diagramas de flujo y qué significa cada uno.
  – Cómo se elaboran los diagramas de flujo.

* Que los alumnos observen experimentalmente qué pasos sigue el ordenador en la ejecución de un programa.

Como sugerencias metodológicas, proponemos las siguientes, relativas a los organigramas de los ejemplos expuestos en este Apéndice:

1. Diagrama para una decisión sencilla (ejemplo 3):

   Un alumno actúa como si fuera el computador.

   Otros alumnos escriben en un papel uno de los dos números o figuras convenidos.

   El profesor simula programar al alumno-computador mediante la instrucción: "Si la papeleta contiene un Ø, escribir un número en la pizarra; si no lo contiene, tirarla a la papelera".

   Entregar la papeleta conteniendo el número puede compararse al hecho de teclear ese número en el ordenador y escribirla en la pizarra obedeciendo la instrucción PRINT.

2. Diagrama para más de una decisión y bucle (ejemplo 4):

   Supongamos que los alumnos pueden escribir en su papeleta un número comprendido entre 1 y 1Ø, ambos inclusive, y que el alumno-computador ha recibido del profesor las siguientes instrucciones:

   1°. Recibir de un compañero una papeleta y leer el número.

   2°. Si el número es par, tirarlo a la papelera y esperar otra papeleta.

   3°. Si el número es impar –es decir, en cualquier otro caso–, compararlo con 5,.y, si es menor o igual, tirarlo a la papelera y esperar otra papeleta.

   4°. En otro caso, escribir en la pizarra el número y esperar otra papeleta.

3. Diagrama para un bucle indefinido hasta que no se cumpla una condición (ejemplo 5):

   Un alumno escribe en la pizarra la tabla de multiplicar por 1 hasta que un com-

pañero le dé, escrito en una papeleta, la letra W, por ejemplo. En ese momento. deja de escribir.
Recordar a los alumnos que la pizarra simula ser la pantalla controlada por el computador, y la papeleta escrita simula ser la tecla gracias a la cual la máquina es instruída.

EJERCICIOS:

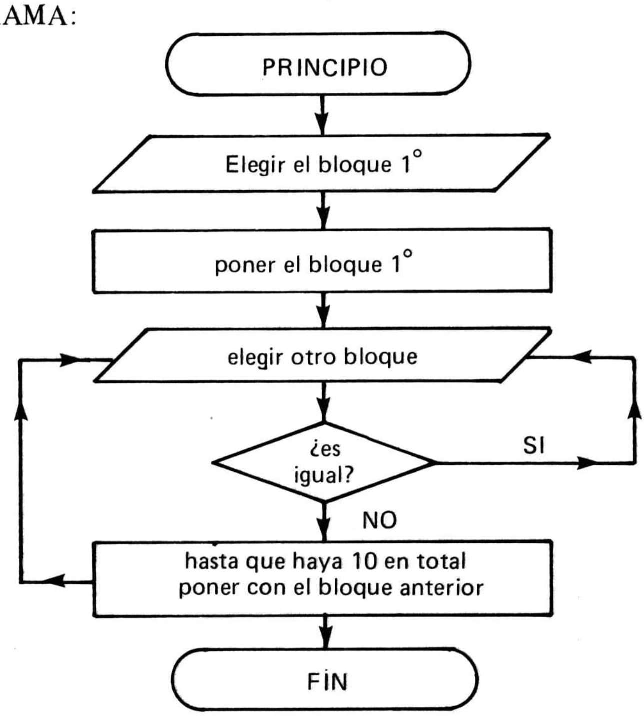
- Confeccionar los diagramas de flujo y desarrollar los programas correspondientes que, una vez ejecutados, impriman en pantalla:
  1. El resultado de multiplicar dos números introducidos por teclado.
  2. "Cuál es la capital de España?"; el nombre de la ciudad introducido por teclado, y "BIEN" si este nombre es "Madrid".
  3. Los números pares introducidos por teclado que sean mayores que 25 y menores que 1ØØ.
  4. Los resultados de multiplicar cualquier número natural por otro número natural menor que 11.
- Confeccionar los diagramas de flujo correspondientes a los programas desarrollados en otros capítulos de este libro.
- Confeccionar diagramas de flujo para desarrollar los propios programas.

  Es conveniente señalar que los alumnos deben iniciarse en la confección de diagramas desde el momento en que empiezan a resolver problemas de cualquier tipo, es decir, desde que inician su escolarización. Unos sencillos ejemplos servirán para aclarar lo que, a simple vista, puede parecer una pretensión utópica.

  1. *Juego para establecer una diferencia entre bloques lógicos:*

Dos alumnos van colocando, alternativamente, un bloque lógico que sólo difiere del anterior por un atributo: tamaño, grosor, color o forma.

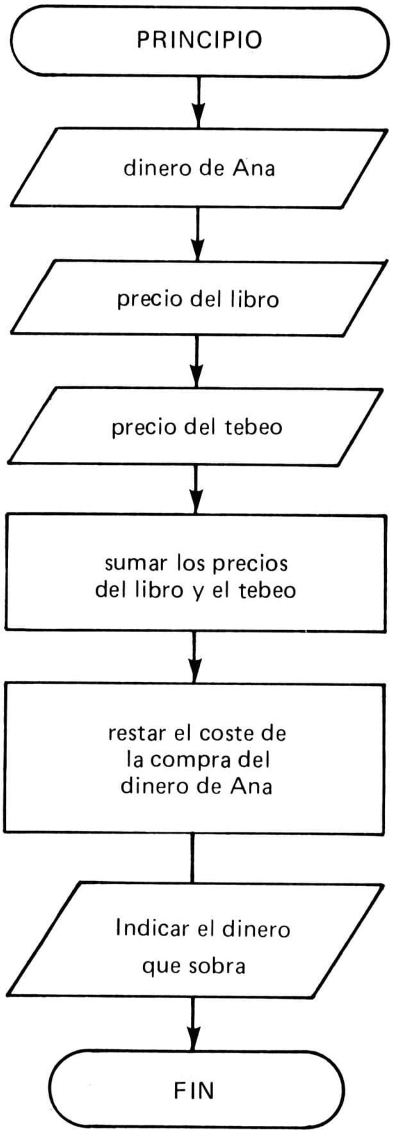
DIAGRAMA:



2. *Problema*

Ana tiene 1.000 pts. y compra un libro por 500 pts. y un tebeo por 125 pts. ¿Cuánto dinero le sobrará?

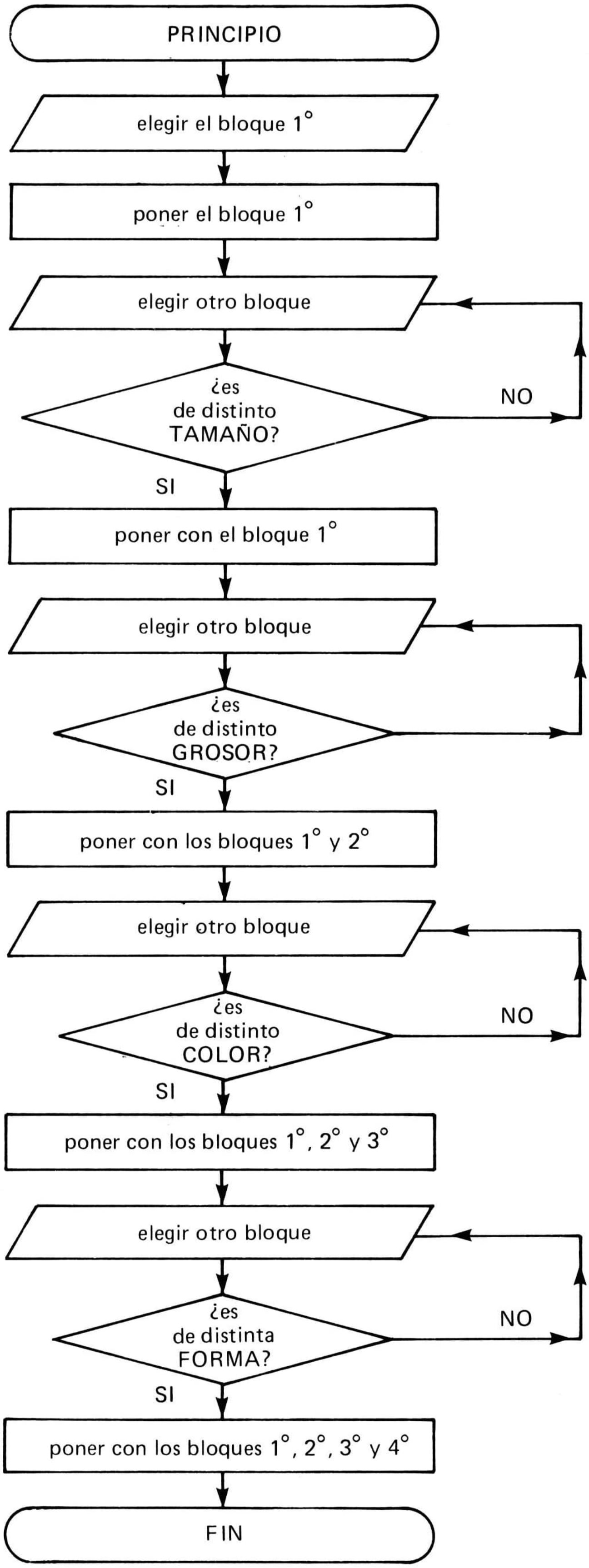
DIAGRAMA:



3. *Juego para establecer una diferencia, por orden, entre bloques lógicos.*

Dos alumnos van colocando, alternativamente, un bloque lógico que sólo difiere del anterior por un atributo: tamaño, grosor, color o forma, sucesivamente.

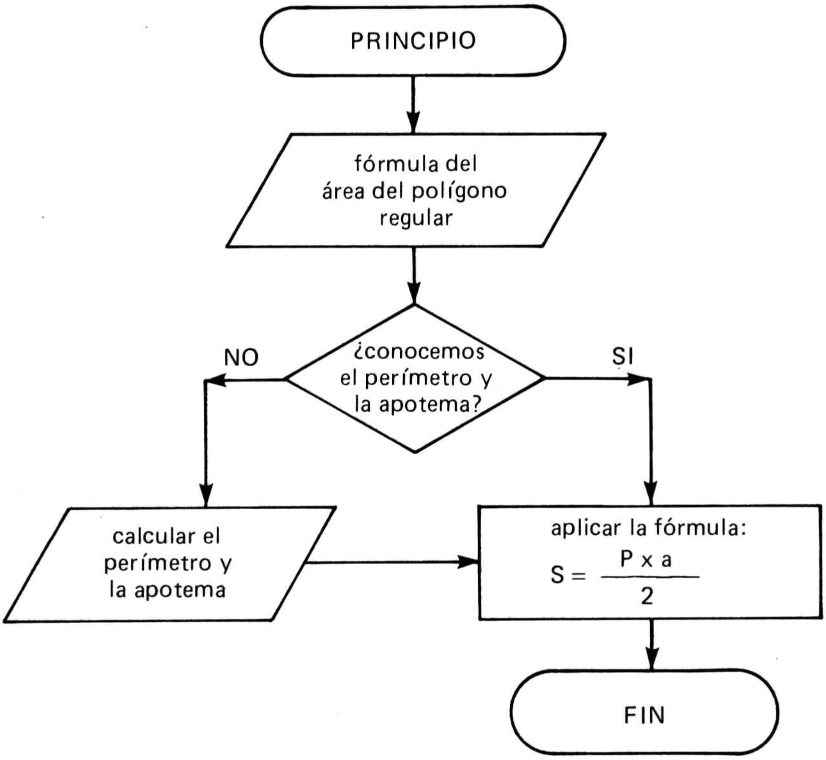
DIAGRAMA:


PRINCIPIO
elegir el bloque 1°
poner el bloque 1°
elegir otro bloque
¿es de distinto TAMAÑO?
NO
SI
poner con el bloque 1°
elegir otro bloque
¿es de distinto GROSOR?
SI
poner con los bloques 1° y 2°
elegir otro bloque
¿es de distinto COLOR?
NO
SI
poner con los bloques 1°, 2° y 3°
elegir otro bloque
¿es de distinta FORMA?
NO
SI
poner con los bloques 1°, 2°, 3° y 4°
FIN

4. *Problema*

Calcular el área de un polígono regular.

DIAGRAMA:



OBSERVACIONES METODOLOGICAS

Los diagramas propuestos en los ejemplos anteriores, por supuesto, no son aplicables directamente al desarrollo de los programas correspondientes. Sería necesario introducir, al menos en algunos de ellos, ciertas modificaciones. Por tanto, deben considerarse como una iniciación a *los diagramas de flujo* en diferentes niveles de enseñanza, ya que cualquier operación lógica puede desarrollarse mediante un proceso capaz de ser reflejado en un diagrama.

Apéndice III ORGANIZACION DE PANTALLAS

Un programa eficaz y bien desarrollado debe incluir una adecuada presentación en pantalla de toda la información que sea necesaria para que el usuario pueda utilizarlo sin dificultad.

Este apéndice está destinado a introducir al lector en *el manejo de la pantalla,* apoyándose en ciertos comandos BASIC y en un mejor conocimiento del ordenador.

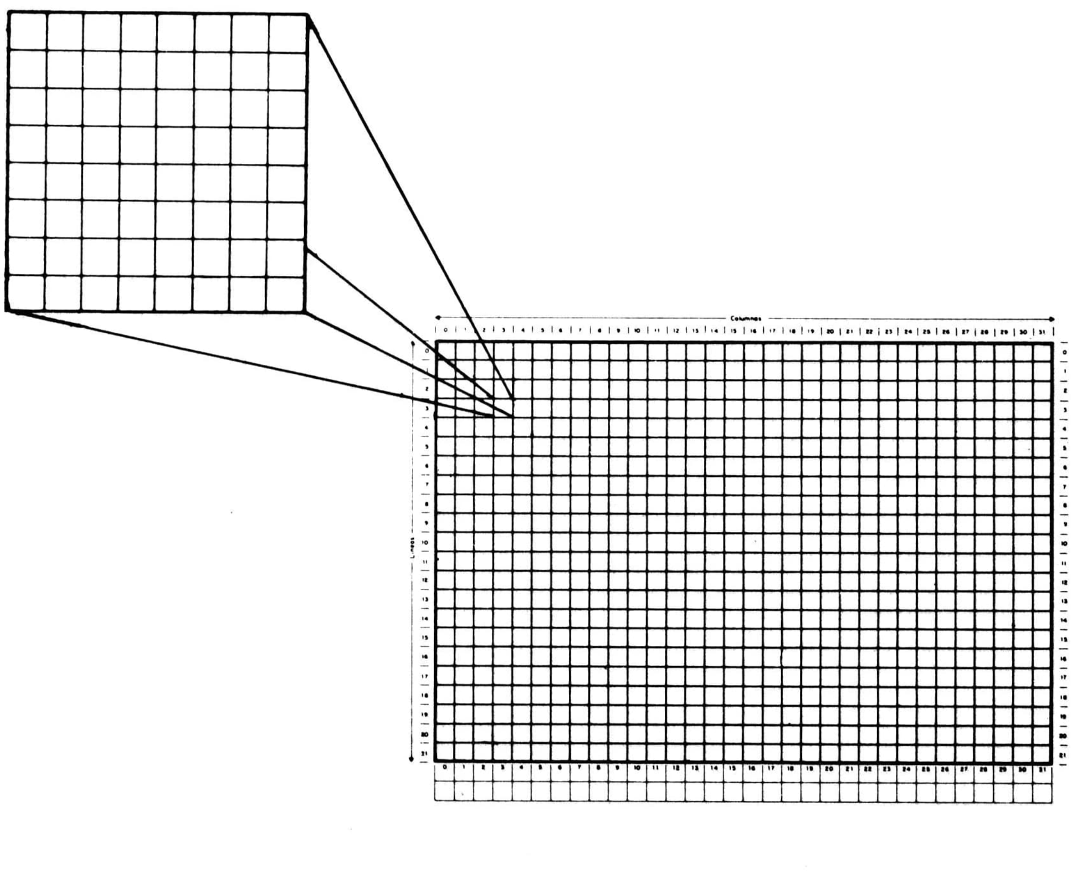
En líneas generales, un computador se comunica con el usuario a través de la pantalla de la unidad de vídeo que tenga conectada y, en sentido contrario, el usuario lo hace gracias al teclado.

Si consideramos la pantalla como una pizarra donde el computador nos va a dar sus respuestas a nuestras cuestiones, debemos empezar por conocer en qué forma la máquina usa esa *"pizarra".*

Cuando se conecta el computador al televisor, el *"cerebro"* de aquél divide la pantalla, de forma inmediata y sin que pueda apreciarse, en una malla o red de **h** cuadritos horizontales y **v** verticales. Las cantidades **h** y **v** son fijas y dependen de cada computador.

Estos cuadritos pueden estar *"encendidos"* o *"apagados"* y, según la combinación de cuadritos encendidos y apagados, se verá en la pantalla una figura u otra. Con esto nos aproximamos a la idea de cómo pinta el ordenador en la pantalla. Veamos ahora cómo *escribe.*

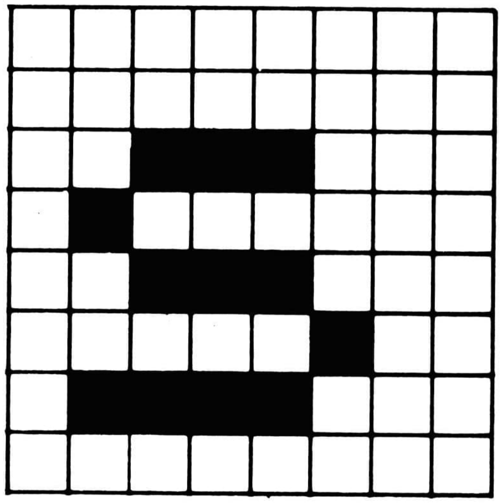
Cada ordenador —de una forma u otra— tiene un conjunto de caracteres que le es propio, tales como todo el abecedario en mayúsculas y minúsculas, los signos de puntuación,



números, etc, y además, los tiene definidos en pequeñas mallas de **m** cuadritos horizontales por **n** verticales, de tal forma que, si tuviera que mostrar en pantalla uno cualquiera de estos caracteres, tomaría, dentro de la malla de **h x v** cuadritos totales, una zona de **n x m** dentro de la cual encendería unos y apagaría otros, monstrando, de esta forma, la figura del carácter deseado.

De lo dicho hasta aquí se desprende que la pantalla gobernada por un ordenador nos puede permitir la impresión de **h/m** caracteres por línea y un total de **v/n** líneas.

Como ya sabemos, con la instrucción PRINT ordenamos a la máquina que imprima en pantalla números y caracteres en general. De acuerdo con lo visto anteriormente, cuando usamos PRINT para escribir un carácter en la pantalla, lo que estamos haciendo en realidad es decirle al computador que nos muestre cuál es el conjunto de cuadritos encendidos y apagados que corresponde a ese carácter y que él tiene *guardado* en su memoria. Veamos, por ejemplo, cuál es la combinación correspondiente a la s:



Estos son los caracteres que están predefinidos en la máquina por su diseñador.

De ahora en adelante, cuando hagamos referencia a caracteres en general, estaremos hablando del espacio ocupado en pantalla por mallas de **n x m** cuadritos.

Un espacio en blanco es una de estas mallas con todos sus cuadritos apagados.

Normalmente, las filas de caracteres, –en pantalla–, van numeradas de arriba abajo, y las columnas, de izquierda a derecha.

Para situar los caracteres en determinada columna, los ordenadores suelen tener la función **TAB**, gracias a la cual podemos imprimir un carácter en la columna que deseemos, dentro de la línea que actualmente esté en servicio. Un ejemplo aclarará esto: Supongamos que queremos imprimir en la columna 5 la palabra **SI** sobre la línea que, según lo impreso con anterioridad, esté disponible. Para ello, nuestro programa podría tener esta configuración:

```
25Ø PRINT "DOS Y DOS"
26Ø PRINT "SON CUATRO?"
27Ø PRINT
28Ø PRINT TAB 5; "SI"
```

Como se puede apreciar, las tres primeras líneas de pantalla están ocupadas de tal forma que, al llegar a la línea 28Ø, el ordenador tiene que escribir en la cuarta, pero además debe obedecer la instrucción **TAB 5**, por la que comenzará a escribir la palabra **SI** a partir de *la columna quinta.*

Este comando es práctico cuando sólo nos interesa controlar la posición horizontal del comienzo de impresión. Si, por el contrario, quisiéramos colocar exactamente en pantalla el primer carácter de la cadena –o número– a imprimir, entonces deberíamos recurrir a un comando del tipo **AT**. Analice el siguiente programa y los resultados de ejecutarlo:

```
1Ø IMPUT "numero de línea"; 1
2Ø INPUT "numero de columna"; c
3Ø PRINT AT 1, c; "*" (1)
4Ø GOTO 1Ø
```

Dé diferentes valores a **l** y **c**.

Hay versiones del BASIC que no tienen los cómodos comandos aquí explicados. Pero en sustitución de los mismos tendrán ciertos códigos de control para *el cursor*, con los cuales se podrán escribir pequeñas *subrutinas* que, finalmente, le permitirán obtener resultados similares a los obtenidos con TAB y AT.

La instrucción PRINT USING es sumamente potente y suelen estar disponibles para computadores de cierta entidad, para los cuales suponemos al lector, a estas alturas de su aprendizaje, capaz de interpretar las explicaciones que el manual de este tipo de máquinas le dé al respecto.

(1) También podemos encontrarnos con PRINT @ cuyo resultado sería el mismo.

Apéndice IV FUNCIONES LOGICAS APLICADAS

Su aplicación a la intrucción IF/THEN

Al hablar sobre **los operadores de relación**, dijimos que estos sólo actúan sobre dos operandos cuyo resultado únicamente puede ser *cierto* o *falso*. Por ejemplo, en **A>B** sólo hay dos posibilidades esperadas: **A** es mayor que **B** o no lo es.

Los operadores lógicos actúan entre *operandos* conformados por *operadores de relación*, lo cual implica que *los operadores lógicos* sólo actúan entre dos posibilidades: *cierto* y *falso*.

En este Apéndice vamos a estudiar la aplicación de **los operadores lógicos AND, OR** y **NOT** en combinación con la instrucción **IF/THEN**.

AND		Y lógico

Analicemos desde un punto de vista estrictamente lógico la siguiente expresión:

IF (A = 15) AND (B > 100) THEN PRINT "bien"

Las posibilidades que se nos ofrecen son que **A** sea o no igual a **15** y que **B** sea o no mayor que **100.** En función de ambas alternativas, escribiremos o no la palabra **"bien"**.

En el ejemplo tenemos dos operadores de relación, "=" y ">", y el resultado de aplicarlos como sabemos entre dos operandos, sólo puede ser *cierto* o *falso*. Asociemos *cierto* con el símbolo **1** y *falso* con el **∅.**

De aquí deducimos que, las combinaciones posibles de **1** y **∅** que nos ofrecen ambas *operaciones de relación* son:

A = B ⇒ cierto ⇒ 1
otra posibilidad ⇒ falso ⇒ ∅
B > 1∅∅ ⇒ cierto ⇒ 1
otra posibilidad ⇒ falso ⇒ ∅

NOTA: El símbolo ⇒ significa "implica".

Volviendo a la expresión inicial, escribiremos las posibilidades de que la palabra "bien" sea escrita:

Si **A** = **B** (cierto ó 1) **Y B** > **100** (cierto ó 1) entonces **sí** se escribe
Si (falso ó Ø) **Y** (falso ó Ø) entonces **no** se escribe
Si (falso ó Ø) **Y** (cierto ó 1) entonces **no** se escribe
Si (cierto ó 1) **Y** (falso ó Ø) entonces **no** se escribe

Es decir que el operador **lógico y** exige que los dos operadores sobre los que actúa sean 1 *–cierto–* para permitir que suceda la acción que le sigue.

Para concluir, hagamos igual a **x** el resultado (Ø ó 1) de la primera operación de relación (A = **15**, en el ejemplo) e igual a **y** el resultado (Ø ó 1) de la segunda operación de relación. (B > **100**, en el ejemplo); pudiendo presentar las posibilidades que ofrece el operador **lógico Y** (AND, en inglés) mediante la siguiente

TABLA DE LA VERDAD: AND

x	y	x AND y	COMENTARIOS
1	1	1	Se permite la acción subsecuente
Ø	Ø	Ø	Se prohíbe la acción subsecuente
Ø	1	Ø	Se prohíbe la acción subsecuente
1	Ø	Ø	Se prohíbe la acción subsecuente

En lo sucesivo nos referiremos directamente a la *tabla de la verdad* de cada operador lógico, evitando la explicación que nos lleva a ella.

EJEMPLO:

Para comenzar los estudios de Bachillerato es necesario tener 14 años y haber obtenido una calificación mínima de 5 puntos en la E.G. Estudie este programa y ejecútelo.

PROGRAMA

```
 1Ø INPUT "Algún alumno?"; A$
 2Ø IF A$ = "sí"  THEN GO SUB 6Ø
 3Ø IF A$ = "no"  THEN STOP
 4Ø RETURN
 5Ø CLS
 6Ø INPUT "Nombre del alumno?; N$: PRINT N$
 7Ø INPUT "Edad?"; E: PRINT E
 8Ø INPUT "Calificación?"; C: PRINT C
 9Ø IF (E>14) AND (C<5) THEN GO TO 14Ø
1ØØ PRINT "Admitido"
11Ø PRINT
12Ø INPUT "Otro alumno?"; A$
13Ø GO TO 2Ø
14Ø PRINT "Rechazado": GO TO 11Ø
```

Este *operador lógico* permite la acción subsecuente siempre que uno cualquiera de los operadores de relación sea **1** (cierto).

Volviendo al ejemplo del operador anterior, pero cambiando éste por el actual **(OR)**, tendremos en términos de BASIC:

IF (A = 15) OR (B > 100) THEN PRINT "bien"

Según la siguiente

TABLA DE LA VERDAD: OR

x	y	x OR y	COMENTARIOS
1	1	1	Se permite la acción subsecuente
Ø	Ø	Ø	No se permite la acción subsecuente
Ø	1	1	Se permite la acción subsecuente
1	Ø	1	Se permite la acción subsecuente

Podemos deducir que la palabra "bien" será impresa en pantalla en el caso de que **A** sea igual a **15 O B** sea mayor que **1ØØ**.

EJEMPLO:

¿Recuerda el programa de Miss Universo? Compárelo con éste, en el que se ha introducido OR. Ejecútelo y observe los resultados.

PROGRAMA

```
1Ø INPUT "Alguna candidata:"; O$
2Ø IF O$ = "sí" THEN GO SUB 6Ø
3Ø IF O$ = "no" THEN STOP
4Ø RETURN
5Ø CLS
6Ø INPUT "Nombre?"; N$: PRINT N$
7Ø INPUT "Talla"; T: PRINT T
8Ø INPUT "Peso?"; P: PRINT P
9Ø IF (T < 1.7) OR (P > 6Ø) THEN GO TO 14Ø
1ØØ PRINT "Admitida"
11Ø PRINT
12Ø INPUT "Otra candidata?"; O$
13Ø GO TO 2Ø
14Ø PRINT "Rechazada": GO TO 11Ø
```

Este es el más sencillo de *los operadores lógicos,* ya que el resultado es falso (Ø) cuando el operador de relación es cierto (1), y cierto (1) cuando el operador de relación es falso (Ø), según la siguiente

TABLA DE LA VERDAD: NOT

x	NOT x	COMENTARIOS
1	Ø	No se permite la acción subsecuente
Ø	1	Sí se permite la acción subsecuente

EJEMPLO:

IF NOT (A = B) THEN PRINT "bien"

Esto implica que la palabra "bien" será impresa en pantalla sólo en el caso de que A no sea igual a B.

Apéndice V TROCEADO DE CADENAS

Por **cadena,** y en términos de programación, entendemos cualquier concatenación de caracteres.

Entre otras cosas, hay que tener en cuenta que una cadena sólo se puede igualar a una variable de caracteres. Tal sería el caso de

A$ = "ROMARBELLAVE, es una ? > 1"

En este ejemplo la cadena está compuesta por letras mayúsculas y minúsculas, dos signos de putuación, un espacio y un par de símbolos matemáticos. En definitiva, caracteres en general.

Trocear una cadena significa subdividirla en cadenas menores. A este tema dedicaremos el Apéndice presente.

Existen diferentes comandos para *el troceado de cadenas,* y éstos varían de un computador a otro. Nosotros estudiamos dos de los más extendidos pero, en todo caso, el lector sacará conclusiones suficientemente provechosas.

Sobre la variable **A$** fijada más arriba, aplicaremos dos comandos de función idéntica pero de léxico y sintaxis distintos, según el computador que se use.

Este sistema lo seguiremos con todas las instrucciones de este apéndice.

Entendemos que, como introducción a esta materia, son suficientes los COMENTARIOS para hacerse idea de la aplicación de cada comando.

Posibilidad 1

Extraer, de la cadena representada por **A$,** una subcadena con los cuatro caracteres situados más a la izquierda.

Comando LEFT$	COMENTARIOS
.... **5Ø LET B$ = LEFT$ (A$,4)** **60 PRINT B$**	La ejecución de estas dos líneas de programa obligará la impresión en pantalla de la cadena ROMA.
Comando TO	
5Ø LET B$ = A$ (1 TO 4) **6Ø PRINT B$**	Idéntico resultado.

Posibilidad 2

Extraer una subcadena de **A$** con los cinco caracteres situados más a la derecha.

Comando RIGHT$

```
........................
5Ø LET B$ = RIGTH$ (A$,5)
6Ø PRINT B$
```

COMENTARIOS

La ejecución de esta dos líneas nos dará en pantalla:

a ? > 1

Comando TO

```
5Ø LET B$ = A$ ( 2ØTO 24)
6Ø PRINT B$
```

Fíjese el lector que el primer carácter de esta subcadena es un espacio.
Idéntico resultado.

Posibilidad 3

Extraer una subcadena de A$ que comience en su tercer carácter y tenga una longitud de 8 caracteres.

Comando MID$

```
5Ø LET B$ = MID$ (A$, 3,8)
6Ø PRINT B$
```

COMENTARIOS

Al ejecutar estas líneas obtendremos:

MARBELLA

Comando TO

```
5Ø LET B$ = A$ (3 TO 1Ø)
6Ø PRINT B$
```

Idéntico resultado.

Si el lector dispone de un determinado computador, debe leer detenidamente el epígrafe que, dedicado a las cadenas, tendrá sin duda el MANUAL de programación del mismo.

Apéndice VI ASCII

American Standard Code for Information Interchange.

El código estándar americano para el intercambio de información es el más extendido entre los fabricantes de ordenadores y es el aplicado, casi sin excepción, por los microordenadores.

Una alternativa la ofrece **IBM** con su **EBCDIC** del cual no hablaremos, pero del que dejamos constancia, dada la amplitud de mercado que abarca esta firma.

El objeto de este Apéndice es introducir al lector en el *porqué* de **la codificación.**

En general, **un código** es una correspondencia biunívoca entre los elementos de un conjunto de valores y otro de símbolos.

De lo anterior se desprende que, para cada carácter —letras, números, símbolos aritméticos, etc— tendremos que establecer un código que permita interpretar, bajo determinadas circunstancias, que un determinado número corresponde a un determinado carácter o símbolo.

Este código debe ser lo suficientemente extenso como para admitir la representación de todos los caracteres necesarios y aún de otros no previstos, y, por otra parte, tan reducido como sea posible para no consumir, sin necesidad, memoria en el ordenador.

Por razones que exceden la intención de este libro, se ha estimado que el número ideal de símbolos susceptibles de entrar en el código es **256.** De esta forma, se cubren todas las necesidades actuales y quedan disponibles codificaciones para símbolos futuros. Este sería el caso de la **ñ** y **los acentos** en español, ya que, al ser el **ASCII** un código diseñado por anglosajones, no han sido necesarios en un principio como símbolos estándar.

TABLA DEL CODIGO ASCII

RESERVADO PARA CODIGOS DE CONTROL Y COMUNICACIONES (filas Ø y 1)

SIN CODIFICAR DE FORMA ESTANDAR (filas 8 a 15)

	Ø	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
Ø	Ø	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
1	16	17	18	19	20	21	22	23	24	25	26	27	28	29	3Ø	31
2	Espacio 32	! 33	" 34	# 35	$ 36	% 37	& 38	' 39	(4Ø	) 41	* 42	+ 43	, 44	– 45	46	/ 47
3	Ø 48	1 49	2 5Ø	3 51	4 52	5 53	6 54	7 55	8 56	9 57	: 58	; 59	< 6Ø	= 61	> 62	? 63
4	@ 64	A 65	B 66	C 67	D 68	E 69	F 7Ø	G 71	H 72	I 73	J 74	K 75	L 76	M 77	N 78	O 79
5	P 8Ø	Q 81	R 82	S 83	T 84	U 85	V 86	W 87	X 88	Y 89	Z 9Ø	[91	\ 92	] 93	Δ 94	— 95
6	' 96	a 97	b 98	c 99	d 1ØØ	e 1Ø1	f 1Ø2	g 1Ø3	h 1Ø4	i 1Ø5	j 1Ø6	k 1Ø7	l 1Ø8	m 1Ø9	n 11Ø	o 111
7	p 112	q 113	r 114	s 115	t 116	u 117	v 118	w 119	x 12Ø	y 121	z 122	{ 123	: 124	} 125	∿ 126	127
8	128	129	13Ø	131	132	133	134	135	136	137	138	139	14Ø	141	142	143
9	144	145	146	147	148	149	150	151	152	153	154	155	156	157	158	159
1Ø	16Ø	161	162	163	164	165	166	167	168	169	170	171	172	173	174	175
11	176	177	178	179	180	181	182	183	184	185	186	187	188	189	19Ø	191
12	192	193	194	195	196	197	198	199	2ØØ	2Ø1	2Ø2	2Ø3	2Ø4	2Ø5	2Ø6	2Ø7
13	2Ø8	2Ø9	21Ø	211	212	213	214	215	216	217	218	219	220	221	222	223
14	224	225	226	227	228	229	230	231	232	233	234	235	236	237	238	239
15	24Ø	241	242	243	244	245	246	247	248	249	25Ø	251	252	253	254	255

Apéndice VII SISTEMAS DE NUMERACION

El único objetivo de este capítulo es recordar **los sistemas de numeración** de forma que, cuando nos refiramos a *binario, decimal* o cualquier otro término de los tratados en este libro, haya un lugar de referencia al que dirigirse.

Antes de entrar en materia, es oportuno recordar unos párrafos de otro libro de esta colección: *"Los Colores y los Gráficos en el Spectrum"*.

"**Digital**".– Perteneciente o relativo a los dedos. Esta es la definición que nos da el diccionario. Pero, ¿cuál es el origen de dicho término?

Los seres humanos tenemos cinco dedos en cada mano, gracias a lo cual resulta normal contar cualquier cantidad como el número de veces que contiene esa cantidad los diez dedos. Así, 24 naranjas corresponden a 2 veces 10 dedos más cuatro dedos. Esto es 2 x 10 + 4 = 24.

24 es un número, y 2 y 4 también lo son, pero los números comprendidos entre Ø y 9 –ambos inclusive– son conocidos como *dígitos* para darnos a entender que son susceptibles de ser contados directamente con los dedos de nuestras dos manos. En otras palabras, *los números dígitos* son los símbolos fundamentales del **sistema decimal** que es, lógicamente, el de uso corriente entre las personas.

En otro orden de cosas, pero profundamente ligado a lo anteriormente expuesto, los ordenadores sólo pueden reconocer si un impulso eléctrico puede o no pasar por un interruptor. Si pasa le asignamos un 1; en caso contrario, un Ø. Vemos pues que un ordenador tiene dos "dedos" y, consiguientemente, solo dos dígitos –o símbolos básicos–, el 1 y el Ø, de un sistema de numeración en **base dos** o **binario.** Esto quiere decir que, al tener dos dígitos, sólo podrá interpretar números formados por esos dos dígitos o, lo que es lo mismo, series de *ceros* y *unos.*

Cualquier número en base diez tiene su equivalente *en base dos* y, lógicamente, al revés. Por ejemplo, 17 en base diez se representa por 10001 *en base dos"*.

Los sistemas de representación numérica y la conversión de números de una a otra base –desde un punto de vista práctico– conforman el contenido de este capítulo.

SISTEMA DECIMAL

Supongamos que tenemos delante de nosotros un montón de cajas conteniendo libros y deseamos conocer cuántos hay, sin saber contar más que con los dedos. Muy probablemente, el procedimiento que seguiríamos sería el siguiente:

Nos situaríamos de tal forma que los libros estuvieran a nuestra izquierda y a continuación tomaríamos una caja y la pondríamos a la derecha. En ese momento nos haríamos una señal en el dedo meñique de una mano, volveríamos a tomar otra caja del montón de la izquierda y la pasaríamos a la derecha, colocándola encima de la anterior, iniciando con esto un nuevo montón, y nos haríamos una señal similar a la anterior en otro dedo. Y así seguiríamos hasta tener todos los dedos de las dos manos con la señal convenida.

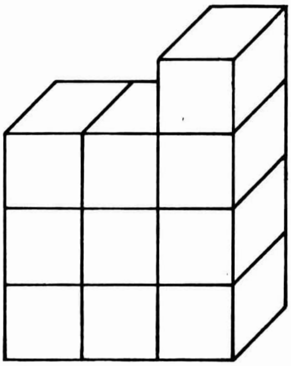
Al llegar a este punto, retiraríamos un poco el montón que hasta el momento hubiésemos acumulado a nuestra derecha; haríamos una raya en el suelo, y borraríamos las señales de nuestros dedos, comenzando el proceso nuevamente con las cajas que aún quedasen a la izquierda.

Y así, una y otra vez, hasta que el montón de cajas inicial hubiese desaparecido.

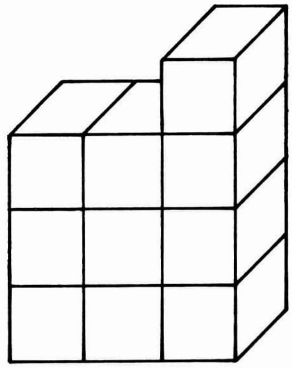
Posiblemente, al finalizar el párrafo anterior, una sonrisa haya aparecido en su cara, pen-

sando en la sencillez del ejemplo. Si es así debe sentirse feliz ya que, como veremos más tarde, ésta es la forma en que un ordenador "cuenta" aunque, desde luego, en binario.

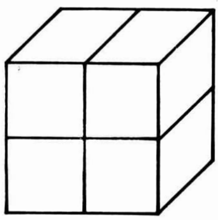
Salvando el paréntesis anterior, sigamos con los montones, que ahora están situados a la derecha, y supongamos que tenemos dos rayas en el suelo —por tanto dos grandes montones de cajas— más cuatro cajas sueltas.



A



B



C

El montón denominado A en el dibujo equivale a 1 vez 10 cajas (*1 decena*).

El montón B equivale a 1 vez diez cajas (*1 decena*).

El montón C es igual a ninguna vez 10 cajas, pero hay físicamente cuatro cajas. Entre los tres montones de cajas tenemos *dos decenas más cuatro.*

En resumen, hay dos veces 10 cajas más cuatro cajas lo cual se puede representar por:

$$10 + 10 + 4$$

o, mejor aún por:

$$2 \times 10 + 4$$

Y esto se ha convenido en representar por sólo los números en negrita, es decir, el **24.**

El número 24 está formado por los dígitos decimales 2 y 4.

SISTEMA BINARIO

Como ya sabemos, **en base dos**, sólo disponemos del **Ø** y el **1** como símbolos básicos o dígitos. Comencemos con el ejemplo anterior.

¿Qué podríamos hacer para contar, en estas circunstancias, las cajas de libros que tenemos a nuestra izquierda?

Antes de dar respuesta a esta cuestión, debemos recordar que los microprocesadores tienen diferente número de **bits,** lo cual significa que pueden manejar números en base dos compuestos por tantos dígitos binarios como **bits** tengan.

Supongamos que nuestro microprocesador tiene *8 bits* y pasemos a ver como *cuenta* nuestro montón de cajas.

Antes de empezar, a la derecha tendríamos:

Ø Ø Ø Ø Ø Ø Ø Ø

vamos, que no hay *ninguna caja.*

Tomamos la primera caja y la ponemos a la derecha, y el número ahora sería:

Ø Ø Ø Ø Ø Ø Ø 1

Hay *una caja.*

Aparentemente nos hemos quedado sin opción ya que se han utilizado los dos dígitos que podemos usar. No obstante, pasemos otra caja a la derecha y reflexionemos.

Cuando en *el sistema decimal* usamos todos los dígitos disponibles –10– hacemos una rayita, y a cada grupo de éstos lo denominamos *diez* o *decena.* En *el sistema binario*, al tener *una pareja*, nos veríamos obligados a hacer una rayita para indicar que hemos cubierto *un par.* Por tanto, nuestro nuevo número binario será:

Ø Ø Ø Ø Ø Ø 1 Ø

Para entendernos, hay un montón a nuestra derecha, formado por dos cajas. Tenemos *un par.* Los separamos un poquito y empezamos otra vez. Tomamos otra caja y la pasamos a la izquierda, y ahora tendremos un montón de dos cajas *–un par–* y otra más, siendo el nuevo binario:

Ø Ø Ø Ø Ø Ø 1 1 – *Un par más uno*

Pasamos otra caja, con lo cual tendremos a nuestra izquierda 2 montones de 2 cajas o, lo que es igual, *2 pares* y el binario correspondiente.

Ø Ø Ø Ø Ø 1 Ø Ø – *Dos pares*

Evidentemente los Ø a la izquierda no son significativos.

Pasamos otra caja y tendremos:

Ø Ø Ø Ø Ø 1 Ø 1 – *Dos pares más uno*

Esto nos indica que a la derecha hay dos montones de dos cajas más una caja o *dos pares más una.*

Con la siguiente caja tendríamos tres montones de dos cajas lo cual implica el siguiente número binario:

Ø Ø Ø Ø Ø 1 1 Ø – *Dos pares más un par*

A continuación

Ø Ø Ø Ø Ø 1 1 1 – *Dos pares más un par·más uno.*

La siguiente caja completaría otro par y, al no haber más combinaciones posibles con los tres dígitos de la derecha, tendremos que pasar a:

Ø Ø Ø Ø 1 Ø Ø Ø – *Cuatro pares*

La siguiente:

Ø Ø Ø Ø 1 Ø Ø 1 – *Cuatro pares más uno*

La siguiente:

Ø Ø Ø Ø 1 Ø 1 Ø – *Cuatro pares más un par*

La siguiente:

Ø Ø Ø Ø 1 Ø 1 1 – *Cuatro pares más un par más uno*

La siguiente:

Ø Ø Ø Ø 1 1 Ø Ø – *Cuatro pares más dos pares*

La siguiente:

Ø Ø Ø Ø 1 1 Ø 1 – *Cuatro pares más dos pares más uno*

La siguiente:

Ø Ø Ø Ø 1 1 1 Ø – *Cuatro pares más dos pares más un par*

La siguiente:

Ø Ø Ø Ø 1 1 1 1 – *Cuatro pares más dos pares más un par más uno*

Como ya no podemos hacer más combinaciones con cuatro dígitos tendremos que pasar a combinaciones con cinco dígitos binarios:

Ø Ø Ø 1 Ø Ø Ø Ø – *Ocho pares*

Y así podríamos seguir:

Ø Ø Ø 1 Ø Ø Ø 1 – *Ocho pares más uno*

Ø Ø Ø 1 Ø Ø 1 Ø – *Ocho pares más un par*

Ø Ø Ø 1 Ø Ø 1 1 – *Ocho pares más un par más uno*

Pudiendo continuar hasta la máxima combinación posible con *8 bits* que corresponde, claro está, a:

1 1 1 1 1 1 1 1

equivalente al decimal 255.

Un poco después, entraremos en la forma en que el microprocesador de *8 bits* obtiene números superiores a éstos.

Ya tenemos una idea de cómo se cuenta en *el sistema binario*. Estudiemos ahora cómo transformar un *número binario* en su equivalente decimal. El procedimiento es muy simple y unos ejemplos lo dejarán claro:

– Transformar el número binario 1 a decimal (1 binario lo representa 1 B)

Procedimiento

$1B = 1 \times 2^0 = 1$

Por tanto, 1 en base 2 (1B) es equivalente a 1 en base 10

– Transformar el 11B a decimal

Procedimiento

$11B = 1 \times 2^1 + 1 \times 2^0 = 2 + 1 = 3$

– Transformar el 101B a decimal

Procedimiento

$101B = 1 \times 2^2 + \emptyset \times 2^1 + 1 \times 2^0 = 4 + \emptyset + 1 = 5$

Transformar el 1 1 1 0 1 1 0 0 B a decimal

Procedimiento

$11101100B = 1 \times 2^7 + 1 \times 2^6 + 1 \times 2^5 + \emptyset \times 2^4 + 1 \times 2^3 + \emptyset \times 2^1 + \emptyset \times 2^9 = 128 + 64 + 32 + 0 + 8 + 4 + 0 + 0 + 0 = 236$

La rutina utilizada se explica por sí misma y es de aplicación para cualquier *número binario* (con ligeros cambios a cualquier número en cualquier base) y responde al *Teorema fundamental de la numeración*, cuya explicación y demostración cae fuera de los objetivos de este libro.

Veamos ahora una forma práctica de utilizar el anterior procedimiento para convertir *números binarios* en *decimales*.

Si confeccionamos una plantilla similar a la que muestra el dibujo:

TABLA 1

128	64	32	16	8	4	2	0

Y, en los cuadros en blanco, colocamos los dígitos del número binario que deseamos pasar a decimal, sólo tendremos que sumar las cantidades que figuran en los recuadros superiores ocupados por *unos* en los cuadros inferiores.

Ejemplo: Siguiendo este procedimiento, pasemos a decimal el número 11101100B.

128	64	32	16	8	4	2	0
1	1	1	0	1	1	0	0

Y ahora sumamos los números decimales de los cuadrados superiores a los que figuran con un 1 binario en la parte inferior, obteniendo de esta forma el decimal buscado:

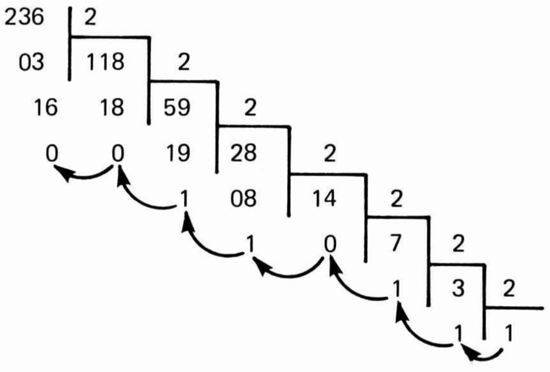
$128 + 64 + 32 + 8 + 4 = 236$

Supongamos ahora que deseamos pasar un *número en base diez* a *base dos.*

El procedimiento es igualmente fácil y un nuevo ejemplo lo aclarará.

– Pasar 236 en base diez, a su equivalente binario

Procedimiento



Si tomamos el último cociente y todos los restos de las divisiones en el sentido de las flechas tendremos el binario buscado:

1 1 1 0 1 1 0 0

SISTEMA HEXADECIMAL

De la misma forma que el sistema decimal tiene diez dígitos y el binario dos, *el sistema hexadecimal* o *en base 16* tiene los siguientes 16 símbolos básicos o dígitos:

0, 1, 2, 3, 4, 5, 6, 7, 8, 9, A, B, C, D, E, F.

Observará que a partir de 9, tenemos que representar los nuevos símbolos básicos por letras mayúsculas del abecedario, para suplir la falta de los mismos.

Si volviéramos al ejemplo del montón de cajas y comenzáramos el proceso de nuevo, los grupos de cajas que, finalmente quedarían a nuestra derecha tendrían que ser, en este caso, de 16 unidades.

Por otra parte y de la misma forma escritos en decimal, contamos:

0, 1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12, 13, 14, 15, 16, 17, 18, 19, 20, 21, 22, ...

en hexadecimal será:

0, 1, 2, 3, 4, 5, 6, 7, 8, 9, A, B, C, D, E, F, 10, 11, 12, 13, 14, 15, 16, 17, 18, 19, 1A, 1B, 1C, 1D, 1E, 1F, 20, 21, 22, ..., 2A, 2B, 2C, 2D, 2E, 2F, 30, 31, ...

La conversión de *números hexadecimales* a *decimales* o *binarios,* y de estas bases a aquéllas, es en todo similar al procedimiento utilizado para convertir *binarios* y *decimales* entre sí.

La única condición es que, allá donde aparezcan los dígitos hexadecimales A, B, C, D, E y F, serán sustituidos por sus valores decimales correspondientes 10, 11, 12, 13, 14 y 15. Unos ejemplos clarificarán los procesos de conversión:

– Pasar al sistema decimal el número hexadecimal 2E

Procedimiento

2E Hex = $2 \times 16^1 + 14 \times 16^0 = 32 + 14 = 46$ Decimal

– Pasar el A3F1 Hex a decimal

Procedimiento

A3F1 Hex = $10 \times 16^3 + 3 \times 16^2 + 15 \times 16^1 + 1 \times 16^0$ = 10.960 + 768 + 240 + 1 = = 41.969 Decimal.

– Pasar 41, 969 decimal a hexadecimal.

Procedimiento

```
41.969 | 16
 099   |2623 | 16
  036    102 | 163 | 16
   049   063   03  | 10
     01 <-- 15 <-- 3 <-'
```

Tomando el último cociente y todos los restos de las divisiones en el sentido de las flechas, tendremos el número hexadecimal correspondiente pero – **¡atención!** – tanto el cociente último como los restos sucesivos, en el supuesto de que sean mayores de 9, deben sustituirse por los dígitos hexadecimales correspondientes.

En el ejemplo que nos ocupa, el último cociente es 10, que equivale al dígito hexadecimal *A*, y el segundo resto es 15 que equivale al dígito *F*. Por tanto, el número en base 16 que buscamos será:

A 3 F 1

De forma similar actuaríamos entre *hexadecimal* y *binario*, pero, no obstante, el siguiente método hará más fácil estas conversiones.

TABLA 2

TABLA DE CONVERSION DE LOS DIGITOS HEXADECIMALES A BINARIO							
0	Ø Ø Ø Ø	4	Ø 1 Ø Ø	8	1 Ø Ø Ø	C	1 1 Ø Ø
1	Ø Ø Ø 1	5	Ø 1 Ø 1	9	1 Ø Ø 1	D	1 1 Ø 1
2	Ø Ø 1 Ø	6	Ø 1 1 Ø	A	1 Ø 1 Ø	E	1 1 1 Ø
3	Ø Ø 1 1	7	Ø 1 1 1	B	1 Ø 1 1	F	1 1 1 1

Supongamos el hexadecimal anterior **A3F1.**

Si de izquierda a derecha, sustituímos cada dígito hexadecimal por el binario correspondiente, dado en la tabla 2, tendremos:

1 Ø 1 Ø	Ø Ø 1 1	1 1 1 1	Ø Ø Ø 1
A	3	F	1

que sería el binario equivalente.

A lo largo de este libro no trabajaremos mucho con números hexadecimales, no obstante, hay ocasiones en que el uso de este sistema de numeración será más práctico que cualquier otro.

TABLA DE CONVERSION DECIMAL Y HEXADECIMAL

Dec.	Hex.	Dec.	Hex.	Dec.	Hex.	2's C.	Dec.	Hex.	2's C.
0	00	64	40	128	80	—128	192	C0	—64
1	01	65	41	129	81	—127	193	C1	—63
2	02	66	42	130	82	—126	194	C2	—62
3	03	67	43	131	83	—125	195	C3	—61
4	04	68	44	132	84	—124	196	C4	—60
5	05	69	45	133	85	—123	197	C5	—59
6	06	70	46	134	86	—122	198	C6	—58
7	07	71	47	135	87	—121	199	C7	—57
8	08	72	48	136	88	—120	200	C8	—56
9	09	73	49	137	89	—119	201	C9	—55
10	0A	74	4A	138	8A	—118	202	CA	—54
11	0B	75	4B	139	8B	—117	203	CB	—53
12	0C	76	4C	140	8C	—116	204	CC	—52
13	0D	77	4D	141	8D	—115	205	CD	—51
14	0E	78	4E	142	8E	—114	206	CE	—50
15	0F	79	4F	143	8F	—113	207	CF	—49
16	10	80	50	144	90	—112	208	D0	—48
17	11	81	51	145	91	—111	209	D1	—47
18	12	82	52	146	92	—110	210	D2	—46
19	13	83	53	147	93	—109	211	D3	—45
20	14	84	54	148	94	—108	212	D4	—44
21	15	85	55	149	95	—107	213	D5	—43
22	16	86	56	150	96	—106	214	D6	—42
23	17	87	57	151	97	—105	215	D7	—41
24	18	88	58	152	98	—104	216	D8	—40
25	19	89	59	153	99	—103	217	D9	—39
26	1A	90	5A	154	9A	—102	218	DA	—38
27	1B	91	5B	155	9B	—101	219	DB	—37
28	1C	92	5C	156	9C	—100	220	DC	—36
29	1D	93	5D	157	9D	—99	221	DD	—35
30	1E	94	5E	158	9E	—98	222	DE	—34
31	1F	95	5F	159	9F	—97	223	DF	—33
32	20	96	60	160	A0	—96	224	E0	—32
33	21	97	61	161	A1	—95	225	E1	—31
34	22	98	62	162	A2	—94	226	E2	—30
35	23	99	63	163	A3	—93	227	E3	—29
36	24	100	64	164	A4	—92	228	E4	—28
37	25	101	65	165	A5	—91	229	E5	—27
38	26	102	66	166	A6	—90	230	E6	—26
39	27	103	67	167	A7	—89	231	E7	—25
40	28	104	68	168	A8	—88	232	E8	—24
41	29	105	69	169	A9	—87	233	E9	—23
42	2A	106	6A	170	AA	—86	234	EA	—22
43	2B	107	6B	171	AB	—85	235	EB	—21
44	2C	108	6C	172	AC	—84	236	EC	—20
45	2D	109	6D	173	AD	—83	237	ED	—19
46	2E	110	6E	174	AE	—82	238	EE	—18
47	2F	111	6F	175	AF	—81	239	EF	—17
48	30	112	70	176	B0	—80	240	F0	—16
49	31	113	71	177	B1	—79	241	F1	—15
50	32	114	72	178	B2	—78	242	F2	—14
51	33	115	73	179	B3	—77	243	F3	—13
52	34	116	74	180	B4	—76	244	F4	—12
53	35	117	75	181	B5	—75	245	F5	—11
54	36	118	76	182	B6	—74	246	F6	—10
55	37	119	77	183	B7	—73	247	F7	—9
56	38	120	78	184	B8	—72	248	F8	—8
57	39	121	79	185	B9	—71	249	F9	—7
58	3A	122	7A	186	BA	—70	250	FA	—6
59	3B	123	7B	187	BB	—69	251	FB	—5
60	3C	124	7C	188	BC	—68	252	FC	—4
61	3D	125	7D	189	BD	—67	253	FD	—3
62	3E	126	7E	190	BE	—66	254	FE	—2
63	3F	127	7F	191	BF	—65	255	FF	—1

Sistemas físicos

¿ORDENADOR O COMPUTADOR?

UNIDADES ELEMENTALES DE INFORMACION

MEMORIA

INTERFACES

PERIFERICOS: Teclado, televisor o monitor, grabador-reproductor de casetes, impresora, discos magnéticos

LOS PROGRAMAS

EL LOGICAL

PROFESOR, ALUMNO Y ORDENADOR

SISTEMAS FISICOS

¿ORDENADOR O COMPUTADOR?

Un computador es una máquina.

Como tal, ha sido diseñada por humanos para ser utilizada por humanos.

Esta premisa es imprescindible para que el lector mantenga una perspectiva adecuada sobre todo lo que sigue.

La función encomendada a esta máquina es manejar información.

Antes de ir más allá, es conveniente conocer el origen de estas dos palabras y su ámbito correcto de aplicación.

Computar significa contar o calcular una cosa por números y esto es lo que hace el dispositivo electrónico al cual nos estamos refiriendo. De aquí se podría inferir que el nombre adecuado debería ser *calculadora;* pero, dado que ya existe éste término para referirnos a ciertos artilugios capaces de realizar —en forma directa— todo tipo de operaciones matemáticas, utilizamos la palabra **computador** para abarcar la idea de *ingenio que, por medio de circuitos lógicos, puede efectuar cualquier operación basada en la de contar, y que opera en el sistema binario de numeración y bajo los criterios del algebra de Boole.*

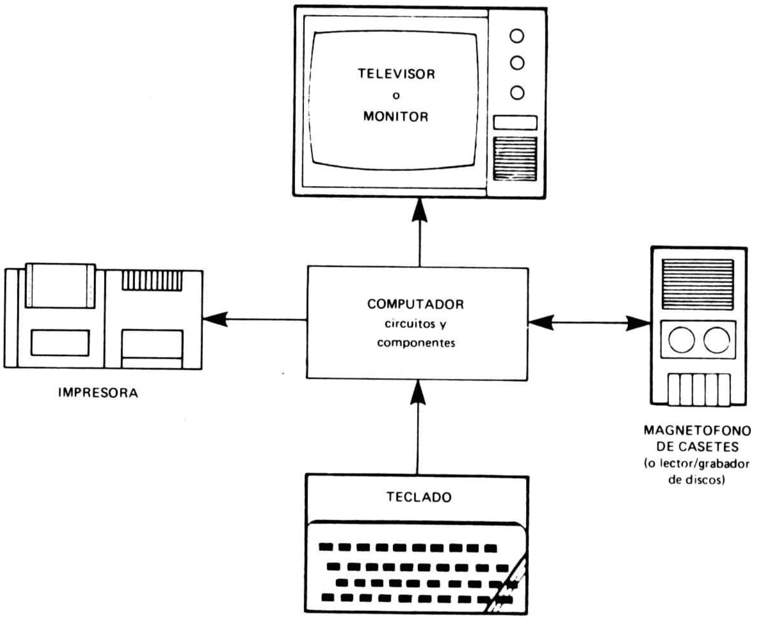
Así pues, un *computador* se limita a operar de acuerdo con la definición anterior, y, consiguientemente, está incapacitado para actuar como una máquina de escribir, o un tubo de rayos catódicos, o un magnetófono ... Pero si está en condiciones de controlarlos o comunicarse con ellos.

Por otra parte, es evidente que, si el *computador* no estuviera habilitado para comunicarse con el exterior, su utilidad sería nula, y, de aquí, la necesidad de unirlo con otros dispositivos a través de los cuales pueda recibir mensajes del exterior (ENTRADAS) y, a su vez, escribir los suyos propios (SALIDAS).

Al conjunto de máquinas electrónicas y/o electromecánicas conectadas y controladas por un computador —incluido él mismo— se denomina **sistema de computación** o, en su acepción francesa, **ordenador.**

Estas puntualizaciones son necesarias para salir del confusionismo existente y poder expresarnos de modo preciso y correcto.

Un esquema típico de *ordenador* es el siguiente:



Las flechas indican el sentido en que circula la información cuando llega el momento de su transferencia entre ordenador y **periférico.**

Periférico es el nombre que, de forma genérica, recibe cualquiera de las máquinas susceptibles de ser conectadas con el computador.

El computador por dentro

Desde el punto de vista social, el coche y el computador tienen muchos puntos en común y estas similitudes serán aprovechadas en ocasiones, a lo largo de este libro, para provocar en el lector ciertas actividades intelectuales necesarias para llegar a dominar este nuevo útil que la tecnología pone a nuestra disposición.

Los mecanismos que componen un vehículo automóvil son muchos y algunos verdaderamente complicados. Pero esto no impide que los podamos agrupar para hacernos una idea de cómo *trabaja* un coche. En este sentido podríamos admitir que existe *un equipo motriz* cuyo fin es dotar a la unidad de fuerza necesaria para mover el resto de los mecanismos y, en definitiva, convertir el conjunto en un *auto-móvil.* De forma similar y adecuada definiríamos *la transmisión, el equipo eléctrico,* etc y, partiendo de estas generalizaciones, podríamos ir conociendo más a fondo los detalles de cada conjunto y la interrelación que los une. Sólo a unos pocos usuarios del automóvil y a los técnicos especializados les podría interesar llegar hasta *la termodinámica* o a *la resistencia de materiales* en el automóvil.

Un moderno **microcomputador** suele estar compuesto por un chasis principal *–circuito principal–* sobre el que están impresos las conexiones necesarias para comunicar electrónicamente todos los componentes *–digitales* o *discretos–* que conforman el computador.

La *fuente de alimentación* no forma parte del computador propiamente dicho, pero es imprescindible para obtener la corriente contínua necesaria, partiendo de la alterna de la red de abastecimiento eléctrico general.

Sobre la tarjeta de circuitos, y de una forma u otra, tenemos la **CPU** o *unidad central de proceso,* la **ULA** u *ordenamiento lógico disponible* (UNCOMUNITTED LOGIC ARRAY), los **chips de memoria,** los circuitos de entrada y salida, y algunos componentes discretos, destinados fundamentalmente a conseguir las tensiones exigidas por los anteriores componen-

tes, denominados **circuitos integrados.**

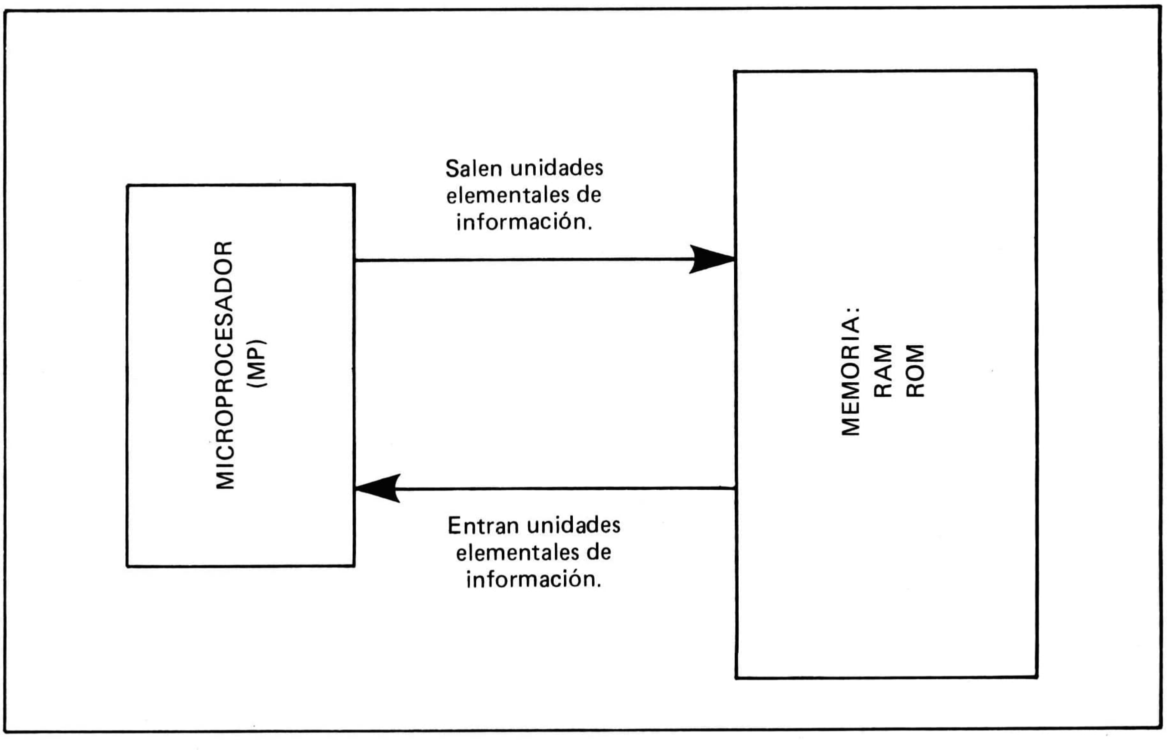
Los circuitos integrados son pequeñas pastillas rectangulares de color negro y con un número variable de patillas, mediante las cuales quedan conectadas al circuito principal. Los circuitos integrados están basados en *la teoría del estado sólido* y pueden llegar a contener el equivalente a miles de componentes discretos del tipo transistor o diodo. También son conocidos por los nombres de **chips** y **cucarachas** en razón de su aspecto externo.

De todos los elementos anteriores, sólo hay dos que realmente nos interesan para aproximarnos a la forma de trabajar de un computador. Estos son: **la CPU** y **los chips de memoria.**

La CPU o *microprocesador* es el verdadero *cerebro* del sistema y en él se producen todas las operaciones de control, comparación y aritméticas que tienen lugar en un computador. Es un circuito integrado de alta densidad.

La memoria de un computador está formada por una serie de *chips* de menor dimensión.

Una disposición física real de la distribución de los diferentes elementos sobre el circuito impreso principal no nos ayudará tanto como el siguiente esquema, en el cual, las flechas nos indican los flujos de información y los bloques las partes fundamentales a estudiar. Su interpretación viene a continuación:



UNIDADES ELEMENTALES DE INFORMACION

La unidad elemental de información que utilizamos al comunicarnos a través de la escritura es la lectura. Nada es más básico.

La electrónica de un ordenador se comunica por medio de su propia unidad elemental de información que es **el byte** (se pronuncia bait).

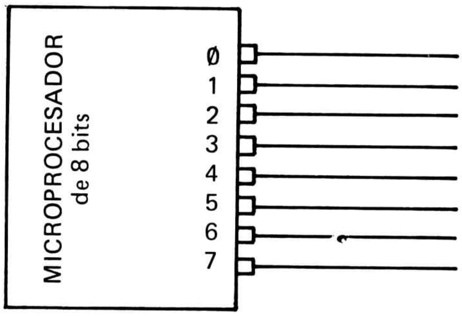
Un *byte* es un número binario o, lo que es igual, cualquier combinación de los dígitos del sistema de numeración en base 2, que son el Ø y el 1, también denominados *bit.*

La palabra **bit** es una contracción de los vocablos ingleses *binary digit* o, en español, dígito binario.

Podemos decir, pues que *un byte es un grupo de bits.*

La razón por la cual el *byte* es el elemento básico de información parte del hecho físico de que un interruptor sólo permite dos posibilidades: dejar pasar corriente o impedirlo.

Si nos referimos específicamente a *un microprocesador de 8 bits,* encontramos que ocho de sus patillas están en condiciones de poder recibir o emitir impulsos eléctricos simultáneamente. Vamos a asociar cada una de estas patillas a Ø, 1, 2, 3, 4, 5, 6, 7.



Supongamos que, por ejemplo, la combinación de impulsos eléctricos que llegan al microprocesador es la siguiente:

7	6	5	4	3	2	1	Ø	← patilla
no lle-ga	lle-ga	lle-ga	lle-ga	no lle-ga	no lle-ga	no lle-ga	lle-ga	← impulso

Esto equivale a decir que *la unidad elemental de información –o byte–* recibida por el microprocesador es Ø 1 1 1 Ø Ø Ø 1 (Ver apéndice "SISTEMAS DE NUMERACION").

Podemos concluir diciendo que el conjunto de operaciones que realiza un computador está basado en dejar pasar o no los impulsos eléctricos. Cuando un impulso pasa, admitimos un 1; en caso contrario, admitimos un Ø.

Las tremendas prestaciones ofrecidas por los computadores se derivan de la rapidez con que manejan enormes cantidades de unidades elementales de información.

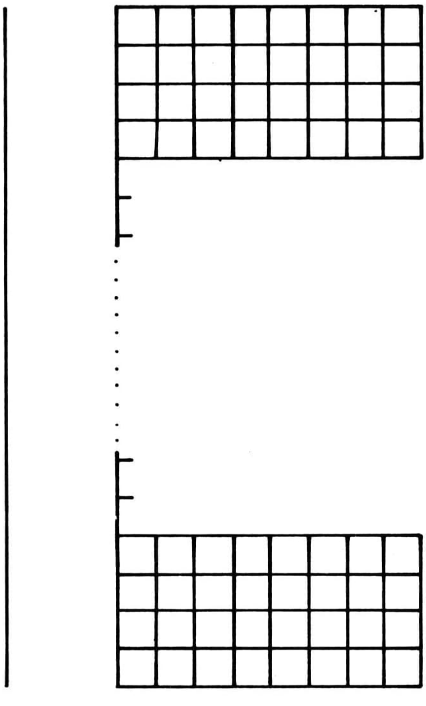
MEMORIA

Las unidades elementales de información son almacenadas dentro de la memoria, quedando a disposición del microprocesador, el cual puede reclamarlas en cualquier momento o depositarlas según sus necesidades.

A cada uno de los lugares donde se puede depositar *un byte* se le conoce como posición de memoria. Todas las *posiciones de memoria* están numeradas de tal forma que podemos referirnos a cualquiera de ellas en función de su dirección de memoria.

Las direcciones de memoria comienzan en la dirección Ø y pueden llegar –para un microprocesador de 8 bits– hasta la 65.535.

Un símil válido podría ser el de una calle con un vecino por casa y todas las casas en la misma acera. En este caso, cada vecino sería un *byte* o *unidad de información.*



Los habitantes de cada casa están representados por diferentes combinaciones de *8 bits.* Es decir, por *un byte.*

Hay muchos computadores que salen de fábrica con algunas de estas posiciones de memoria grabadas indeleblemente y, por tanto, el contenido de las mismas no puede ser alterado, sirviendo al microprocesador sólo para *leer* su contenido y, en ningún caso, para escribir en ellas.

A la memoria de estas características se la denomina **ROM** (Read Only Memory) o **memoria de sólo lectura.**

Al resto de la memoria, sobre la cual, indistintamente, se puede *leer* o *escribir*, se la conoce como memoria **RAM** (Random Acces Memory) o **memoria de acceso aleatorio.**

La *memoria RAM* se borra –pierde el contenido de sus posiciones de memoria– en el momento en que se desconecta el computador. De aquí su calificativo de "volátil".

INTERFACES

Un interface tiene por misión conseguir el entendimiento necesario entre dos máquinas electrónicas.

Como sabemos, un ordenador está compuesto por *un computador* y otros dispositivos, a los cuales controla y con los que está conectado.

Gracias a los *interface* se consigue la conexión física y la transmisión adecuada de unidades elementales de información entre el *computador* y todos *los periféricos* que le sirven.

El aspecto externo de *los interface* es muy variopinto, pero, desde el punto de vista de la comunicación, dos son las técnicas que el lector debe conocer.

En este sentido, los *interface* se diferencian en el modo de efectuar la transmisión de las unidades elementales de información. **El modo paralelo,** envía todos los bits de un byte a un tiempo, mientras que **el modo en serie** lo hace uno tras otro.

Hay dos tipos de *interface* para comunicaciones en *paralelo:* **Centronics** e **IEEE 488.**

El tipo *Centronics* es muy utilizado para la conexión entre *impresora* y *computador,* y es muy raro que, de una forma u otra, los computadores no lleven incorporado un tipo u otro de *interface* paralelo.

Para la transmisión *en serie* se utiliza, generalmente, la norma **RS-232-C** (también conocida en telecomunicaciones como **V-24**) y en ella es fundamental la velocidad de transmisión, debiendo estar sincronizada entre computador y periférico.

La velocidad de transmisión se mide en **baudios** y corresponde al número de *bits* que se emiten o reciben por segundo.

Con esta visión de conjunto sobre un sistema de computación, vamos a aproximarnos a algunos *periféricos.*

PERIFERICOS

El teclado es, por excelencia, el dispositivo de ENTRADA de información, ya que, gracias a él, el computador recibe de forma directa las órdenes del usuario.

El teclado es un dispositivo destinado exclusivamente a *la entrada de información.* Con frecuencia, forma parte de la caja que recubre el computador y la manera de usarlo depende del tipo de ordenador al que va incorporado, siendo, por tanto, imprescindible atenerse a las instrucciones que al respecto dé el MANUAL del mismo.

El televisor o **monitor,** según los casos, es un dispositivo exclusivamente de salida de información.

El televisor como dispositivo de salida y *el teclado* como dispositivo de entrada, permiten una interactividad total entre el usuario y su ordenador.

Es evidente que, dado que el computador sólo es capaz de operar sobre bytes, existirá una codificación-decodificación para que el usuario pueda comunicarse de una forma práctica con el sistema. En otras palabras, el ordenador recibe y presenta mensajes en caracteres entendibles por los humanos, pero trabaja internamente con *bytes.* (Ver Apéndice ASCII).

El grabador-reproductor de casetes es utilizado como medio para la conservación de información en una casete. Si la información parte del computador, entonces queda almacenada en la cinta magnética que está en la casete. Si la información parte de la cinta magnética, ésta pasa a la memoria RAM del computador.

La cabeza magnética del magnetófono se encarga de las transformaciones adecuadas entre impulsos electrónicos y estados magnéticos.

La impresora es un dispositivo para la salida de información que permite al usuario proveerse de copias —impresas en papel— de listados de programas, datos o reproducciones del contenido de la pantalla.

Las impresoras se conectan con el computador a traves de un interface tipo Centronics o de un RS-232, proveyéndonos de copias sobre papel de la información contenida en el ordenador.

Antes o después, todo usuario de un microordenador sentirá la necesidad de proveerse de *una impresora,* encontrándose con un mundo de posibilidades aparentemente contradictorias. El objetivo de estas líneas es guiarle en los primeros pasos de la toma de decisión sobre la elección de la máquina que, según el tipo de aplicación, le pudiera convenir.

La idea que debe prevalecer sobre cualquier otra es que el precio –elevado o bajo– no significa nada a la hora de tomar esta decisión. Esto quiere decir que el usuario debe definir la función que ha de cumplir la impresora y no rechazar sistemáticamente las baratas en favor de las caras o, en sentido contrario, obstinarse en una barata sin considerar que una de precio más elevado es la adecuada, aunque ello signifique un esfuerzo económico adicional.

Impresoras térmicas

Son las más económicas en su adquisición y mantenimiento; son silenciosas y aceptablemente rápidas en la impresión; tienen la gran ventaja de que no necesitan cintas para imprimir sobre el papel, con lo cual –mientras haya papel en *el cargador*– cumplen su función, y no son propensas a las averías, ya que tienen pocos mecanismos.

Su gran inconveniente es la mayor o menor abundancia de papel térmico en el mercado: hay que asegurarse de la existencia del papel adecuado por parte del vendedor.

La técnica de impresión de estas máquinas consiste en *"quemar"* el papel térmico con la cabeza de impresión, obteniéndose de esta forma la figura de los caracteres. Por esta razón no deben dejarse las copias obtenidas sobre papel térmico expuestas al sol, ya que desaparecería la impresión por *"quemado"* del resto de la capa térmica del papel.

Por todas las razones expuestas, estas impresoras son muy adecuadas para trabajos de desarrollo y gabinete.

Impresoras sobre papel normal

Abarcan todos los campos de aplicación, siendo las características que debe observar el potencial comprador las siguientes:

1. Velocidad de impresión.
2. Dispositivo de impresión: agujas o caracteres.
3. Dispositivo de arrastre del papel: Tracción o fricción.

La velocidad de impresión, según el tipo de aplicación, puede ser determinante. Por ejemplo, si se necesita una gran cantidad de cartas personalizadas en un tiempo reducido. Este factor encarece el precio, de tal forma que este sube espectacularmente a medida que aumenta la velocidad.

El dispositivo de impresión sólo debería ser motivo de consideración si la calidad de impresión prevalece sobre cualquier otro, ya que las impresoras de caracteres son lentas con respecto a las de agujas, obteniéndose a cambio un tipo de impresión igual al de una máquina de escribir.

Finalmente, *los dispositivos de arrastre* a utilizar en la impresora son de diferentes tipos.

Los de tracción exigen un papel perforado en sus bandas y contínuo para permitir que un engranaje lo haga avanzar en el carro.

Los de fricción permiten la introducción de hojas sueltas normales, efectuando el avance del papel gracias a la presión que ejerce el rodillo de la impresora sobre el mismo.

Un dispositivo no excluye el otro, siendo ideal la impresora que ofrece ambas posibilidades.

Es evidente que, para determinados trabajos, puede ser muy interesante la presentación sobre cuartillas o folios, mientras que en otras ocasiones la utilización de papel contínuo es imprescindible por rapidez, eficacia y comodidad.

Es el ususario, en definitiva, quien tiene que combinar precio, prestaciones y campo de aplicación de la impresora para obtener la solución más económica.

Los discos magnéticos son soportes de información cuya misión, en todo y por todo, es similar a la que cubren las cintas de casete y que es, en definitiva, el almacenamiento de estados magnéticos, los cuales se transforman –mediante los dispositivos adecuados– en impulsos eléctricos, que a su vez son representativos de unidades elementales de información.

El magnetófono es el artifico electromecánico encargado de esta conversión cuando trabajamos con casetes. La casete, por su propia configuración, sólo permite que la cinta que contiene sea desplazada hacia adelante o hacia atrás, o, lo que es igual, no podemos acceder a un lugar determinado de la cinta de modo directo. Por eso, las cintas se llaman *soportes de información de acceso secuencial.*

Los discos magnéticos son manejados por artificios parecidos a los tradicionales tocadiscos de audio, pero, mientras ésos sólo "leen" el contenido de las pistas de sus discos, aquéllos "leen" y "escriben".

En inglés el nombre que reciben es el de **disk-drive**, que podríamos asimilarlo al popular concepto de tocadiscos.

Existen dos tipos de discos, en función de su rigidez: **floppy** y **duros.**

Los discos duros van incorporados de forma fija al tocadiscos de tecnología Winchester, permitiendo, el conjunto, capacidades de almacenamiento superiores a los **5M bytes** (1 Megabyte = 1000 K) y con una velocidad de giro de unas 3500 vueltas por minuto.

El término "*floppy*" hace referencia a un disco con cierta flexibilidad, que se introduce en el tocadiscos correspondiente cuando es necesario. Nos referiremos a ellos como discos, sin más.

Las medidas estándar para los discos son 8 pulgadas y 5 pulgadas y 1/4. Pronto se impondrán los discos de medidas situadas alrededor de las tres pulgadas.

LOS PROGRAMAS

Desde el punto de vista informático, **un programa** es una lógica y ordenada sucesión de instrucciones dirigidas a hacer operativo un computador.

En otras palabras, un computador es absolutamente incapaz hasta que un programa le indica *qué debe hacer* y *cómo debe hacerlo.*

El hecho mismo de que existan instrucciones dirigidas al computador exige, implicitamente, la necesidad de una forma de comunicación entre el hombre y la máquina (**interlocutores emisores y receptores**).

La forma de comunicación requiere, a su vez, un *medio dísico* (**canal**) para transmitir y un *sistema* (**código**) para distinguir unas instrucciones de otras.

El medio de comunicación con la máquina puede ser cualquiera de los dispositivos de entrada ya conocidos.

El sistema de comunicación nos lleva –por analogía– a establecer lenguajes sometidos a similares normas que los impuestos a los idiomas humanos. Son conocidos como **lenguajes de programación.**

Estos *lenguajes de programación* tienen, como es lógico, **un vocabulario** y **una sintaxis.** En función de la comodidad de su *sintaxis* y la facilidad de interpretación de su *vocabulario,* encontraremos desde lenguajes exhaustivos hasta otros altamente desarrollados.

Para aclarar este punto, conviene recordar que los computadores sólo reconocen series de impulsos, que son, en definitiva, *unidades elementales de información.*

Si las instrucciones llegan al microprocesador desde el computador *byte* a *byte*, estamos programando de forma exhaustiva y directamente en la jerga que entienda la máquina y, por ende, esta forma de programar se denomina **código máquina.**

La expresión *código máquina* es tan clara que podríamos, para reforzar ideas posteriores, llamarla *"código no humano"*. Y hasta tal punto es así, que no es usual ni aconsejable.

Partiendo del *código máquina* se han confeccionado lenguajes más y más desarrollados, cada una de cuyas instrucciones la podemos concebir como un pequeño programa en *código máquina* diseñado de tal forma que permite una fácil comunicación con el computador.

Este tipo de lenguajes son conocidos como *de alto nivel* y requieren, como es lógico, un intérprete para convertir sus instrucciones del lenguaje *de alto nivel* en instrucciones elementales.

Cuanto más exhaustivo sea un lenguaje de programación, más rápidamente ejecutará el microprocesador sus instrucciones, ya que más se aproxima al código máquina, e, inversamente, cuanto más desarrollado, es más lento —en términos relativos—, puesto que necesita la traducción del *intérprete*.

Estos *lenguajes de alto nivel* no representan especial dificultad a la hora de estudiarlos y, sin duda, una persona llega a comunicarse con cierta soltura con el computador en menos tiempo y con menos esfuerzo que si pretendiera un resultado equivalente con un idioma extranjero.

No obstante, es conveniente advertir al lector, previniendo eventuales desfallecimientos, que una pequeña dificultad se va a interponer constantemente en su camino de aprendizaje: **el idioma.**

Los *lenguajes de alto nivel* han sido desarrollados por angloparlantes y, consiguientemente, sus vocabularios están formados por palabras inglesas o contracciones de ellas. Por esta razón, recomendamos que la primera aproximación que se realice a cada palabra del vocabulario del lenguaje que se estudia sea, no ya una mera traducción de la misma al español, sino una asimilación de la imagen mental que, para un inglés, puede tener la acepción aplicada al caso.

Veamos, como ejemplo, el caso de la palabra STR$ del lenguaje de alto nivel BASIC —del cual tratamos ampliamene en otro lugar—. Esta es una contracción de STRING, cuyo significado en español vendría a ser *"cuerda"*. *"hilera"*, etc., y el símbolo de dólar ($), que figura al final —como ya tendremos oportunidad de estudiar— viene a indicar carácter.

Si encajamos ambas imágenes, tendríamos *"cuerda o hilera de caracteres"*; la idea que trata de transmitir es una serie de caracteres puestos uno tras otro y sólidamente unidos, algo así como *una cadena de caracteres.*

Nosotros, al estudiar el BASIC, haremos especial hincapié en la estructura mental que envuelve cada palabra.

EL LOGICAL

Con este sustantivo hacemos referencia al conjunto de programas que, de una forma u otra, permiten a los ordenadores convertirse en eficaces.

La expresión inglesa equivalente es **software.**

Podemos utilizar esta expresión para hacer referencia a todos los programas existentes, sea cual sea su propósito. También podemos aludir a la biblioteca de programas —¿programoteca?— que una persona tenga. Usaremos la palabra **logical,** igualmente, para abarcar los programas que un ordenador tenga incorporados de forma permanente.

El logical para aplicaciones concretas es el que más interesará al usuario. Así, a un profesor le atraerá el *logical educativo* y, probablemente, otros de utilidad para su labor docente personal.

El logical comprende, por supuesto, aquellos programas que cada usuario desarrolla por sí mismo y para su propio uso.

La más dura de las colonizaciones que podríamos soportar proviene de la no generación de *un logical* adaptado a nuestra cultura.

Un computador está capacitado para manejar números y caracteres. En razón de esto, podemos obtener una principal división del *logical* en **procesos de textos, procesos numéricos** y **bancos de datos.**

Mediante *los programas de proceso de datos,* conseguimos convertir un ordenador en algo que va más allá de la más perfecta máquina de escribir electrónica con secretaria incluída. Pero esto no implica que la necesidad de un mecanógrafo desaparezca; muy al contrario, dentro de los conocimientos de un profesional de esta área, se requerirá el manipulado de *un proceso de textos.*

En todo caso, *un tratamiento de textos* permite al usuario de un ordenador:

- Suprimir letras, palabras y párrafos.
- Reemplazarlos, desplazarlos, repetirlos, ajustarlos, etc.
- Y una vez dado por bueno un texto, guardarlo en memoria –externa y permanente–; recuperarlo en todo momento, modelar en pantalla la presentación definitiva de lo escrito en cuanto a espaciados, márgenes y tipos de letras, para, finalmente, imprimir sobre papel y, de forma automática, tantas copias como se quiera cambiando de una a otra, por ejemplo, el encabezamiento.

Estas maravillas del *logical* se encuentran en el mercado por precios desde unas 10.000 pts.

Los procesos numéricos permiten a los más modestos computadores tranformarse en auxiliares matemáticos de primera magnitud para cualquier profesional. Resuelven un algoritmo con prontitud y precisión, ya sea un complicado cálculo de laboratorio, el seguimiento de una contabilidad o la evolución estadística de un grupo de alumnos en un curso.

Por último, **los bancos de datos** están constituídos por grupos de información selectiva y ordenada según unos criterios de jerarquía, pudiendo el usuario acceder a ellos de acuerdo con las normas impuestas por el logical que lo maneja.

Los bancos de datos privados –cada usuario de ordenador puede confeccionar el suyo– pueden contener desdes resúmenes de artículos de prensa, hasta extractos del curriculum personal de los alumnos de un centro escolar.

PROFESOR, ALUMNO Y ORDENADOR

Antes, la relación alumno-profesor sólo era concebible en forma directa.

Ahora la posibilidad de que el profesor se apoye en este auxiliar ha aparecido, modificando las relaciones tradicionales según este esquema:

PROFESOR ↔ ALUMNO
PROFESOR ↔ ORDENADOR
ALUMNO ↔ ORDENADOR

La relación clásica entre un profesor y sus alumnos no varía, excepto por el hecho de que aquél debe enseñar a éstos a manejar su auxiliar: *el ordenador.*

La relación profesor-ordenador es, con mucho, la más dura de las tres posibilidades ofrecidas y pasa por dos etapas bien definidas.

La primera requiere del profesor un esfuerzo para dominar la máquina y hacerla pasar de un elemento pasivo a componente dinámico de sus clases.

La segunda etapa exigirá del profesor un constante ejercicio de imaginación para que la tercera relación propuesta (alumno-ordenador) logre sus objetivos didácticos diariamente.

La relación alumno-ordenador debería basarse en una menor *presión* por parte del profesor y una total interactividad del estudiante con la máquina, de forma que las correcciones sobre los fallos cometidos y las evaluaciones de los ejercicios que realice sobre el computador sean un asunto confidencial del estudiante con la máquina. En nuestra opinión, el profesor debe dejar a *su auxiliar* hacer su "trabajo", reservando para sí el fin último de aumentar el nivel de conocimientos de sus alumnos.

Ofrecemos finalmente un ejercicio práctico que servirá para disponer de un programa que permita el control de la Biblioteca del centro escolar.

```
   1 REM Ficha:" DIM "
   5 REM "Si graba datos debe comenzar en la linea 15"
  10 DIM L$(100,20): DIM A$(100,20): DIM A(100)
  15 CLS : PRINT "BIBLIOTECA"
  20 PRINT ,1;"-ENTRADAS",,2;"-SALIDAS";,,3;"-INVENTARIO"
  25 PRINT ,4;"-FIN TRABAJO"'''"Elija un numero de menu"
  30 IF INKEY$>"0" THEN  IF INKEY$<"5" THEN  GO TO 50
  40 GO TO 30
  50 LET A=VAL INKEY$: GO TO 1000+A*100
1100 CLS : FOR N=1 TO 100
1110 IF CODE L$(N)<>32 THEN  GO TO 1170
1130 PRINT "ENTRADAS"''"Nombre del libro:"': INPUT L$(N)
1140 IF CODE L$(N)=32 THEN  GO TO 15
1160 PRINT N;"-";L$(N): PRINT "Importe: ";:
1162 INPUT A(N): PRINT A(N)
1165 LET A$(N)="BIBLIOTECA"
1170 NEXT N
1180 CLS : PRINT "MEMORIA AGOTADA": PAUSE 0: GO TO 15
1200 CLS : PRINT "SALIDAS"''
1205 PRINT "No. de Libro:": INPUT N
1210 PRINT L$(N)''"En poder de: ";A$(n)''
1215 PRINT "Lector:"'': INPUT C$: IF LEN C$=0 THEN  GO TO 15
1220 PRINT C$: LET A$(N)=C$
1230 GO TO 1205
1300 CLS : PRINT "INVENTARIO"
1310 LET I=0: FOR N=1 TO 100
1315 IF CODE L$(N)=32 THEN  GO TO 1360
1320 PRINT N;"  ";L$(N)
1330 PRINT "    Lector:",A$(N)'"    Importe:",A(N)
1340 LET I=I+A(N)
1350 NEXT N
1360 PRINT '"TOTAL VALOR",I
1370 PAUSE 0: GO TO 15
1400 CLS : PRINT "FIN TRABAJO"'''''''
1410 PRINT "Si desea grabar los datos,"''
1415 PRINT "esciba ""GRABAR"" y pulse ENTER"
1420 INPUT c$: IF c$<>"GRABAR" THEN  GO TO 15
1430 SAVE "BIBLIOTECA"
```

BASIC PARA NIÑOS. Watt y Mangada. 128 páginas. 5ª edición.

Es un libro para niños. Y también es un libro para aquellos padres que quieren acompañar a sus hijos en el aprendizaje del lenguaje BASIC.

No son necesarios conocimientos previos sobre programación. A lo largo del texto, numerosas notas aclaratorias, comentarios y guías didácticas permiten explicar las dudas que pueden surgirle al niño.

Todos los procesos y programas se analizan detalladamente.

De forma amena, utilizando ejemplos sencillos y relacionados directamente con el entorno infantil, se cubren los primeros pasos de programación en BASIC, facilitando una sólida base para el estudio posterior de desarrollos más complejos. Es pues, una obra ideal para niños.

BASIC AVANZADO PARA NIÑOS. Watt y Mangada. 160 páginas. 2ª edición.

— **¿Qué es el código ASCII?**

— **¿Cómo y cuándo utilizar CHR$?**

— **¿Para qué sirven LEN, MID$, LEFT$, RIGHT$?**

— **¿Qué es eso de los operadores lógicos AND y OR?**

Este libro responde a éstas y otras muchas cuestiones de la programación en BASIC. Y lo hace de una manera clara y sencilla, de forma que cualquier niño podrá —utilizando su ordenador— desde organizarse su propio fichero de direcciones, hasta listar alfabéticamente una relación nominal, pasando por autoevaluar sus conocimientos escolares u organizar su propia liga de baloncesto...

Apenas son necesarios unos mínimos conocimientos de programación. A lo largo del texto, numerosas notas aclaratorias, comentarios y guías didácticas explican las dudas que puedan surgirle al niño y amplían sus conocimientos.

Utilizando ejemplos sencillos y directamente relacionados con su entorno, este libro confiere al niño un sólido conocimiento de la programación en BASIC.

BASIC PARA ESTUDIANTES. Antonio Bellido y Arsenio Sánchez.

El libro que permite seguir un curso de BASIC, dirigido bien por Profesor o de forma autodidacta, por estudiantes de los últimos cursos de EGB, BUP, COU, FORMACION PROFESIONAL y otros estudios.

Eminentemente práctico con numerosos ejercicios que al mismo tiempo de hacerlo ameno permite ejercitar la programación.

ENSEÑANZA ASISTIDA POR ORDENADOR. Burke. 1985.

Esta publicación tiene por objeto la familiarización del lector con el desarrollo y creación de lecciones a través del método EAO (CAI), íntimamente relacionado con la utilización de los microordenadores. Incluye materias de gran utilidad en áreas tales como el adiestramiento en el diseño de validación, el perfeccionamiento de las lecciones y el control de calidad de un curso.

BASIC PARA MAESTROS

Hasta hace pocos años, la utilización de los ordenadores en el campo de la enseñanza estaba limitado, por sus elevados costes y mantenimiento, a grupos muy reducidos. En cambio, hoy, con la aparición de los microordenadores, su relativamente bajo coste en el mercado y, como consecuencia más inmediata, su difusión, resulta forzosa la incorporación de la Informática a los centros escolares con tres objetivos fundamentales:

1. Como poderoso auxiliar en la gestión del propio centro.
2. Como medio para guiar, apoyar y reforzar el aprendizaje de las direferentes materias.
3. Como materia fundamental, por sí misma, del currículo escolar.

Esta publicación, dirigida a los profesores interesados en Informática, que carecen de conocimientos de la misma o los poseen muy elementales, pretende introducirlos en ella a través del lenguaje de programación BASIC, eliminando, en todo lo posible, las dificultades de aprendizaje y aportando sugerencias didácticas que posibiliten su aplicación en las aulas.

Con este libro:

- Aprenderá el LENGUAJE de programación BASIC.
- Aprenderá cómo ENSEÑAR BASIC a sus ALUMNOS.
- Tendrá cientos de PROGRAMAS RESUELTOS.
- Conseguirá una guía complementaria de INTRODUCCION A LA INFORMATICA.

EDITORIAL PARANINFO S.A.
Magallanes, 25 - 28015 Madrid

ISBN: 84-283-1375-X